

# Energy-Efficient and Secure System Architecture for Wireless Sensor Networks

**Habilitationsschrift**

Submitted to attain the degree of Venia Legendi.

Department of Computer Science

University of Augsburg

Dr. Faruk Bagci



## Abstract

Wireless Sensor Networks (WSN) have emerged as a new information-gathering paradigm based on the collaborative efforts of a large number of self-organized sensing nodes. These networks form the basis for many types of smart environments such as smart hospitals, intelligent battlefields, earthquake response systems, and learning environments. A set of applications, such as biomedicine, hazardous environment exploration, environmental monitoring, military tracking and reconnaissance surveillance, are the key motivations for the recent research efforts in this area. Different from traditional networks, sensor networks do impose a set of new limitations for the protocols designed for this type of networks. Devices in sensor networks have a much smaller memory, constrained energy supply, less process and communication bandwidth. Topologies of the sensor networks are constantly changing due to a high node failure rate, occasional shutdown and abrupt communication interferences. Because of the nature of the applications supported, sensor networks need to be densely deployed and have anywhere from hundred to thousands of sensing devices, which are orders of magnitude larger than traditional ad hoc mobile networks. In addition, energy conservation becomes the center of focus because of the limited battery capacity and the difficulty of recharge in the hostile environment. With fundamental difference between traditional networks and sensor networks, it is not appropriate and probably inefficient to port previous solutions for ad hoc networks into sensor networks with only incremental modifications. For ensuring the functionality of a sensor network, especially in malicious environments, security mechanisms are essential for all sensor networks. However, sensor networks differ from classical (wireless) networks and this consequently makes it harder to secure them. Reasons for this are resource constraints of the sensor nodes, the wireless multi-hop communication, and the possibility of node compromise. Since sensor nodes are often deployed in unattended or even hostile environments and are usually not equipped with tamper-resistant hardware, it is relatively easy to compromise a sensor node. By compromising a sensor node, an adversary gets access to all data stored on the node, such as cryptographic keys. This thesis discusses concepts and mechanisms to cope with energy efficiency and security issues in wireless sensor networks. The contribution of this work is to provide a novel system architecture for WSN which offers interfaces for diverse applications to directly control communication, topology, mobility, security, localization, and energy-saving based on predefined requirements. Each of these features are interleaved with each other, e.g. changing topology parameters would directly influence energy-saving. Communication manages the local access to the medium by neighboring sensor nodes in an efficient way. In this thesis, we analyze different MAC protocols and

develop a dynamic approach combining contention based and time slot based communication schemes. We investigate several network topologies and provide a flexible way to build hierarchical structures in energy efficient way. The variable range protocol adjusts communication range of each sensor node to a minimum without losing overall network connectivity. The mobility feature allows applications to update or reprogram sensor nodes in a convenient way. This thesis provides a tiny mobile agent system to change or modify applications. The main focus of the thesis lies on securing data and communication in WSN. We introduce a architecture that offers security features in multiple communication levels. The security goes hand in hand with energy saving to compensate the increased performance need of encryption in communication. This thesis investigates also localization issues in WSN. We develop a location estimation and tracking system using wireless sensors. All these features combined in one middleware form a novel energy-efficient and secure system architecture for wireless sensor networks.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Outline . . . . .	5
<b>2</b>	<b>Wireless Sensor Networks</b>	<b>7</b>
2.1	Sensor Node Technology . . . . .	7
2.1.1	Hardware . . . . .	9
2.2	Applications of WSN . . . . .	12
2.3	Characteristics of WSN . . . . .	14
<b>3</b>	<b>Related Work</b>	<b>17</b>
3.1	Energy Efficient Communication . . . . .	17
3.2	Code Mobility in WSN . . . . .	19
3.3	Location Estimation and Tracking . . . . .	20
3.4	Security in WSN . . . . .	22
<b>4</b>	<b>Energy-Efficient Communication</b>	<b>27</b>
4.1	The ESTR Protocol . . . . .	28
4.1.1	Network Structure and Configuration . . . . .	28
4.1.2	Node Communication . . . . .	30
4.1.3	Self-optimizing by Dynamic Sleeping . . . . .	32
4.2	Evaluation . . . . .	33
4.2.1	Average Energy Consumption . . . . .	33
4.2.2	Communication Latency . . . . .	38
4.2.3	Real-life Tests . . . . .	39
<b>5</b>	<b>Variable Ranges Protocol</b>	<b>43</b>
5.1	Network Establishment . . . . .	44
5.2	Probabilistic Routing in VR Protocol . . . . .	46
5.3	Evaluation . . . . .	47
<b>6</b>	<b>Mobility in WSN</b>	<b>53</b>
6.1	UbiMASS Architecture . . . . .	54
6.1.1	UbiMASS ELF Loader . . . . .	55
6.1.2	Migration Engine . . . . .	57
6.1.3	Sensor-Actuator Interface . . . . .	59

6.1.4	Communication Layer . . . . .	60
6.1.5	UbiMASS Agent . . . . .	61
6.2	Evaluation . . . . .	62
<b>7</b>	<b>Security Architecture for WSN</b>	<b>67</b>
7.1	Security Requirements in WSN . . . . .	67
7.2	Security Threats in WSN . . . . .	70
7.3	Security Attacks in WSN . . . . .	70
7.3.1	Layering-Based Security Attacks in WSN . . . . .	71
7.3.2	Routing Mechanism Attacks in WSN . . . . .	75
7.4	SecSens - A Novel Security Architecture for Wireless Sensor Networks	76
7.4.1	Authenticated Broadcasts . . . . .	77
7.4.2	Key Management . . . . .	78
7.4.3	Routing . . . . .	81
7.4.4	En-route Filtering . . . . .	83
7.4.5	Evaluation . . . . .	84
<b>8</b>	<b>Location Estimation and Tracking in WSN</b>	<b>87</b>
8.1	LocSens - Indoor Location Tracking Using Wireless Sensors . . . . .	88
8.2	Evaluation . . . . .	91
8.2.1	First Optimizations . . . . .	93
8.2.2	Location Tracking . . . . .	94
8.3	Optimizations for LocSens . . . . .	95
8.3.1	Cluster optimization . . . . .	96
8.3.2	Optimization of Room Sensor Positions . . . . .	98
8.3.3	Optimization of Location Tracking . . . . .	100
<b>9</b>	<b>Conclusion</b>	<b>103</b>



# 1 Introduction

After the invention of the transistor in the mid 50's mainframes were only accessible to research institutions. The progressive miniaturization and the decline in prices for computer components leveraged personal computers to their triumph. Meanwhile, the trend goes away from wired desktop to much smaller, but still powerful computer systems such as smart-phones, laptops, or net-books. This development also led to a useful and interesting area: the sensor networks.

Sensors are a quite old technology. Galileo Galilei wrote in his letters to a friend about a device to measure heat in the 17th century. It was probably the first thermometer. Electronic sensors arose from the technological development. Today, sensors are used nearly in every manufacture to monitor and control the production. It is unimaginable to build cars, airplanes, and all kinds of machines without sensors. Technological progress allows more and more sensors to be manufactured on a microscopic scale as micro-sensors using MEMS technology.

Sensor networks consist of many small, inexpensive but highly energy-efficient microcomputers with the purpose of sensing and monitoring a certain environment. Depending on the application, the monitored environment can be covered by hundreds or even thousands of sensor nodes. Sensor nodes consist of a sensing, a processing, a transceiver, and a power unit. The sensing unit is used to perform some measurements of some physical phenomena, e.g., temperature, pressure, movements. The processing unit consists of a slow CPU and a restricted memory space. It enables a sensor node to pre-process the measured data before sending it using the transceiver unit. The measurements are sent to a central base station called sink. The power unit usually consists of a battery pack. In addition, a sensor node may be equipped with some additional application-dependent components such as location finding system, actuators etc.

Most WSN applications scenarios, especially large-scale WSNs with thousands of sensor nodes, require that an individual sensor node has to be cheap. As a result, the available resources are extremely constrained. The constrained computational power of a sensor node requires the use of only efficient operations which are suitable for the limited CPU and the restricted memory space. Especially the limited energy resources make energy saving of paramount importance to achieve a long lifetime of a WSN. Thus, the overhead for computation and communication must be low. Since communication requires a significant amount of energy, only



as few messages as necessary which are as short as possible should be transmitted. Since the range of the wireless transceiver is also limited, messages are sent in a multi-hop communication, i.e., sensor nodes forward messages hop by hop to the destination.

A distributed system architecture for WSNs should offer applications the possibility to interface different parameters in order to influence communication, energy-efficiency, and security of the network. Figure 1.1 illustrates a possible system architecture for wireless sensor networks.

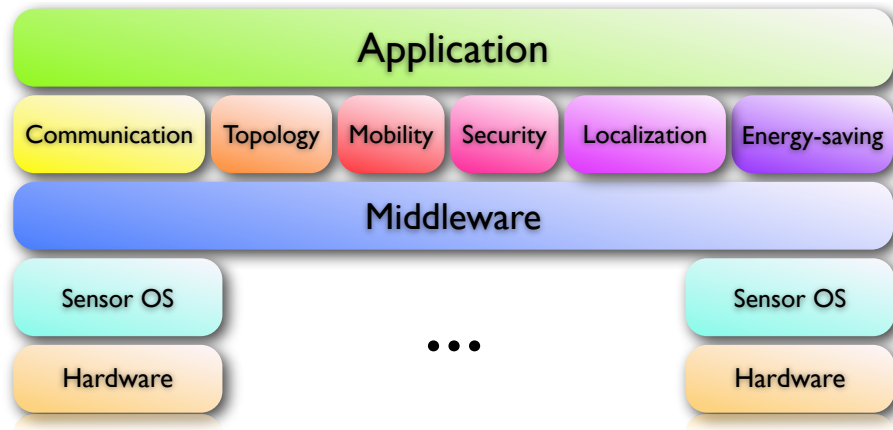


Figure 1.1: Distributed System Architecture for Wireless Sensor Networks

On top of the architecture the application level is located. Based on application requirements the system architecture can focus on different features, i.e. communication, topology, mobility, security, localization, and energy-saving. Features are interconnected with each other, e.g. changing parameters in topology would directly influence energy-saving.

- Communication aims the local access of the medium by neighboring sensor nodes in an efficient way. In general, there are contention based and time based communication schemes. Both schemes have their own advantages and disadvantages. In this thesis, we propose a mixed usage of contention and time based communication.
- A network can be build using different topologies. Small adjustments in the topology of a network can effect highly energy consumption and security. A system architecture should offer applications to dynamically adapt topology to changes in network structure.
- The mobility feature offers users to easily update or reprogram sensor nodes in the network. Code mobility is a convenient approach to change or modify applications.

- In most sensor projects, security is disregarded because of complexity and performance reasons. Nevertheless, the system architecture should provide possibilities to secure data and communication in applications.
- Location of a node is probably the most important context in wireless sensor applications. Especially, determining the exact position of a node indoors is very complicated. Therefore, the system architecture should provide functionalities for location estimation and tracking of sensor nodes. In all features, energy-saving is an important optimization goal.

The main focus of this thesis is to elaborate energy efficient and secure communication in a parallel way. A middleware abstracts from diverse sensor nodes that are distributed in the environment. A sensor operating system, in some cases also named firmware, is used to access node hardware components, like micro-processor, memory, sensors, and transceivers. The middleware offers applications interfaces to modify parameters on nodes for local communication, topology, mobility, localization, security, and energy saving.

## 1.1 Outline

This thesis is organized as follows. In Chapter 2, we present necessary background information of wireless sensor networks including WSNs characteristics, architecture, communication systems, topologies, classifications, and applications. Chapter 3 gives an overview about related projects in the areas of energy efficient communication, code mobility, location estimation and tracking, and security in WSN. The following chapters describe features of the proposed system architecture and present evaluation results for developed approaches. Chapter 4 introduces an energy efficient protocol for local communication in WSN. Topology aspects are discussed in chapter 5 and a novel protocol is described to dynamically adapt connections in the network with the aim to optimize energy consumption. Chapter 6 elaborates code mobility issues in WSN and introduces a mobile agent system for sensor networks. A novel security architecture for WSN is the focus of chapter 7. In chapter 8 location estimation and tracking in WSN is discussed and an indoor location tracking system is introduced. The thesis finishes with the conclusion.



## 2 Wireless Sensor Networks

The convergence of the Internet, communications, and information technologies, coupled with recent engineering advances, is paving the way for a new generation of inexpensive sensors and actuators, capable of achieving a high order of spatial and temporal resolution and accuracy. The technology for sensing and control includes sensor arrays, electric and magnetic field sensors, seismic sensors, radio-wave frequency sensors, electro optic and infrared sensors, laser radars, and location and navigation sensors.

Advances in the areas of sensor design, materials, and concepts will further decrease the size, weight, and cost of sensors and sensor arrays by orders of magnitude and will increase their spatial and temporal resolution and accuracy.

The technology for sensing and control now has the potential for significant advances, not only in science and engineering, but equally important, on a broad range of applications relating to critical infrastructure protection and security, health care, the environment, energy, food safety, production processing, quality of life, and the economy. In addition to reducing costs and increasing efficiencies for industries and businesses, wireless sensor networking is expected to bring consumers a new generation of conveniences, including, but not limited to, remote controlled heating and lighting, medical monitoring, automated grocery checkout, personal health diagnosis, automated automobile checkups, and child care.

In this chapter we will explore many aspects of WSNs environment such as sensor node technology, overview of wireless ad hoc networking as a base of WSN, description of several WSNs perspectives (WSNs characteristics, architecture, communication systems, topologies, and classifications), routing protocols in WSNs, and finally WSNs applications.

### 2.1 Sensor Node Technology

Recent advances in Micro Electro Mechanical Systems (MEMS), tiny micro-processors and low power radio technologies have created low-cost, low-power, multi-functional miniature sensor devices (nodes), which can observe and react to changes in physical phenomena of their surrounding environments.

Wireless sensor networks (WSN) combine three basic features in a single system: sensing, processing, and wireless communication. The idea is not only to measure a physical quantity, but also to process this real world data locally and to send gathered information to interested entities. All of these functions are kept in a single device called sensor node. A sensor node is mainly composed of four basic components: a power unit, sensing unit, processing unit, and transceiver (transmit/receive) unit. It also may contain some additional components to serve some applications like position finding, and an actuator as shown in Figure 2.1.

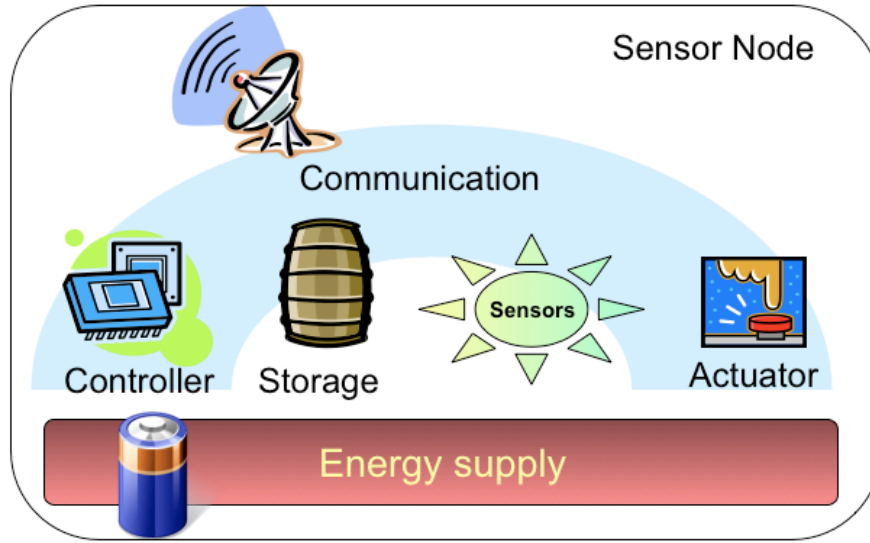


Figure 2.1: Basic sensor node components

By means of the transceiver circuitry, a sensor unit communicates with nearby units relying on RF communication. Currently available sensors employ one of two types of radios. The simplest (and cheaper) alternative offers a basic Carrier Sense Multiple Access (CSMA) Medium Access Control (MAC) protocol, operates in a license free band (315/433/868/916 MHz) and has a bandwidth in the range 20 - 115 kbps. Newer models support an 802.15.4 radio operating in the 2.4 GHz band and offering a 250 kbps bandwidth. The radio range varies with a maximum of about 300 meters for the first radio type and 125 meters for the 802.15.4 radios [BPC<sup>+</sup>07].

In general, a sensor node is suffering from limited resources as follows [WLSC06]:

- *Limited Memory and Storage Space:* A sensor node is a tiny device with only a small amount of memory and storage space, with such a limitation, the software built for the sensor node must also be quite small.
- *Power Limitation:* Energy is the biggest constraint to wireless sensor ca-

pabilities. Once sensor nodes are deployed in an environment, they cannot be easily replaced (high operating cost) or recharged (high cost of sensors). Therefore, the battery charge taken with them to the field must be conserved to extend the life of the individual sensor node and the entire WSN.

- *Unreliable Communication:* Certainly, unreliable communication is a major threat to WSN security. The security of the network relies heavily on a well defined security protocol which in turn depends somewhat on communication.

### 2.1.1 Hardware

Sensor hardware can be divided in four groups [HHKK04]. *Special-purpose sensor nodes* are purposely designed to sacrifice flexibility in order to be as small and inexpensive as possible. In contrast, *generic sensor nodes* can be flexibly adjusted to different application scenarios. They provide a rich expansion interface where various different types of sensors can be attached. Typical examples are the MICA Motes [HSW<sup>+</sup>00] [HC02] or the TelosB motes [PSC05]. *High-bandwidth sensor nodes* possess processing and communication capabilities to deal with complex sensor streams, including video and voice processing. An example are the Bluetooth equipped BTnodes [BTn07]. *Gateway nodes* can act as a sink providing the link between the WSN and backbone infrastructure, i.e., the network connection to the task manager node and the user. The Stargate platform is a typical example for gateway nodes [Cro07].

Hardware platforms for WSNs are developed by many research groups and commercial companies. Since WSNs are currently in a development stage and there is no true killer application that would decrease the costs, there exists no platform which dominates the market. For many research groups it is often more convenient and even less expensive to build their own WSN devices instead of buying commercial ones. However, the above mentioned Berkeley MICA Motes and their clones are broader used as other platforms and are accepted as the de facto platform in the research community. Algorithms and protocols proposed for WSNs usually assume hardware resources comparable to the MICA Motes.

Figure 2.2 shows a MICA2 Mote. The sensor node is equipped with an Atmel ATmega128L processor which is based on a Harvard RISC architecture. The maximum clock signal of the CPU is 16 MHz which provides a throughput up to 16 MIPS. The MICA2 mote provides 128 kbyte Program Flash Memory, 512 kbyte Measurement Flash, and 4 kbyte Configuration EEPROM. The radio is a TI CC1000 (formerly Chip-Con) with a data rate of 38.4 kbps. The standard packet size is 36 byte (with a payload of 29 byte). The successor MICAz has similar properties except that the data rate is increased to 250 kbps and the

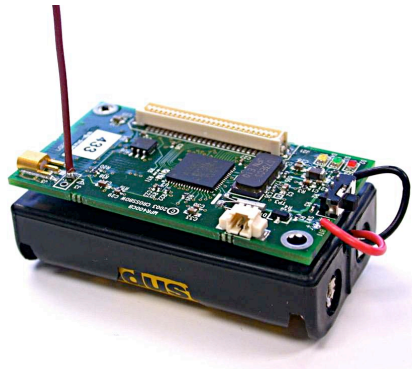


Figure 2.2: Berkeley MICA2 Mote

packet size can be up to 128 byte using the ZigBee transceiver CC2420. ZigBee is a short-range, low-power, low-cost, low-data-rate wireless multihop networking technology standard. It is built upon the Physical (PHY) and Medium Access Control (MAC) layers of the IEEE 802.15.4 standard [IEE06]. ZigBee specifies network and application layer, as well as the security service provider. MICA2 and MICAz nodes are powered with two AA batteries. The energy requirements of the MICA2 motes are stated differently in the literature. To save energy, sensor nodes can use different sleep modes.

Another hardware platform called Sun SPOT [Sun] is developed by Sun Microsystems. The sensor nodes are equipped with a 180 MHz 32-bit ARM920T processor with 512 kbyte RAM and 4 Mbyte Flash-memory. The radio is a TI CC2420 and is IEEE 802.15.4 compliant. An USB interface is also present. A Sun SPOT node is powered with a 3.7V, 750 mAh lithium-Ion battery that is rechargeable via the USB interface. Sun specifies the operation time up to 7 hours with both CPU and the radio active. By having the processor in sleep mode and by turning off the radio the operation time is extended. Figure 2.3 shows a Sun SPOT node.

Although these nodes have different characteristics, their basic hardware components are the same. Therefore, these hardware components should be organized in a way that makes them work correctly and effectively without a conflict in support of the specific applications for which they are designed. Each sensor node needs an operating system (OS) that can control the hardware, provide hardware abstraction to application software, and fill in the gap between applications and the underlying hardware.

Traditional OS functions are therefore to manage processes, memory, CPU time, file system, and devices. This is often implemented in a modular and layered fashion, including a lower layer of kernels and a higher layer of system libraries. Traditional OSs are not suitable for WSN because they have constrained re-



Figure 2.3: Sun SPOT Mote

sources and diverse data-centric applications, in addition to a variable topology. To support the node operation, it is important to have open-source operating systems designed specifically for WSNs. Such operating systems typically utilize a component-based architecture that enables rapid implementation and innovation while minimizing code size as required by the memory constraints endemic in sensor networks. Such operating systems should embody the following functions, bearing in mind the limited resource of sensor nodes:

- Compact and small in size since the sensor nodes have very small memory. The sensor nodes often have memories of only tens or hundreds of kilobytes.
- Provide real-time support, since there are real-time applications, especially when actuators are involved. The information received may become outdated rather quickly. Therefore, information should be collected and reported as quickly as possible.
- Provide efficient resource management mechanisms in order to allocate microprocessor time and limited memory. The CPU time and limited memory must be scheduled and allocated for processes carefully to guarantee fairness (or priority if required).
- Support reliable and efficient code distribution since the functionality performed by the sensor nodes may need to be changed after deployment. The code distribution must keep WSNs running normally and use as little wireless bandwidth as possible.
- Support power management, which helps to extend the system lifetime and improve its performance. For example, the operating system may schedule the process to sleep when the system is idle, and to wake up with the advent of an incoming event or an interrupt from the hardware.



- Provide a generic programming interface up to sensor middleware or application software. This may allow access and control of hardware directly, to optimize system performance.

TinyOS is one such the most famous of a de facto standard, but not the only one [Uni]. It is written in nesC, a programming language for deeply networked systems [GLvB<sup>+</sup>03]. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools; these can be used as-is or be further refined for a specific application. TinyOS's event-driven execution model enables fine-grained power management.

## 2.2 Applications of WSN

Typical applications of WSNs include monitoring, tracking, and controlling. Some of the specific applications are habitat monitoring, object tracking, nuclear reactor controlling, fire detection, traffic monitoring, etc. In a typical application, sensor nodes are scattered in a region where they are meant to collect data and send it to the sink node. The main applications of WSNs can be categorized as follow:

- *Military applications:* In military applications such as battlefield surveillance, target tracking and monitoring, enemy reconnaissance, etc., WSNs can be deployed over a region where some phenomenon is to be monitored. As an example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion.
- *Environmental monitoring:* Many valuable applications can be found in agriculture where sensor networks could be fitted with a near infinite variety of chemical and biological sensors. A simple but very useful application is the control of irrigation. In large farms and ranches that may cover several square kilometers only portions of the area may get some rain sporadically. Knowing which areas of the field need irrigation can direct the soil watering system to precisely irrigate those areas. Such an application is ideal for wireless sensor networks since data rate over the network is very low and delay of several minutes or hours could be tolerated.
- *Medical applications:* With the development of biological sensors, health monitoring is expected to grow quickly. The development of these sensors might experience a breakthrough if conventional CMOS integrated circuits can be combined with biological sensors. Applications in this area can be the monitoring of the blood sugar and blood pressure, heart pulses and respiration rate. Some sensors might just be wearable and attached externally others might be implanted to allow the monitoring of the person. In

a first stage these sensor networks might find their application area on big farms where sensors will be attached to the animals. The sensor information might be used for location information or for determining the need for treatments to prevent parasites.

- *Engineering applications:* WSNs can also be deployed for many applications such as power plant monitoring, oil pipeline monitoring, fault detection in bridges or dams, building monitoring, etc.
- *Commercial applications:* WSNs can be used for environmental control in industrial and office buildings, inventory control, vehicle tracking and detection, traffic flow surveillance, etc.
- *Industrial automation:* Industrial facilities consist of a set of relatively large physical plants where long cables need to be installed to transfer the state of the plant to the control room (valves, temperature, pressure, etc.) which is centralized in most facilities. Significant cost savings can be achieved when these cables are replaced by inexpensive wireless communication. Because most of the information being communicated is state information, it only changes relatively slowly, therefore in normal operation traffic is very low but when it is required to send new state information, the network must be very reliable. Exactly this could be guaranteed by a wireless sensor network where every node has many neighbors so that many different routing paths would exist from source to destination. A further example is the use of wireless sensor networks on moving or rotating machinery where wired sensors are simply impractical.
- *Home applications:* include home automation, instrumented environment, and automated meter reading.

As this wide variety of different applications shows, WSNs may have totally different properties and characteristics. WSNs may vary from small networks to large networks consisting of hundreds or thousands of sensor nodes. The used sensor node hardware may vary from tiny, battery-powered and extremely resource constrained sensor nodes to powerful sensor nodes with permanent energy supply that are able to perform extensive computations. Just as well, the WSN can consist of different types of sensor nodes where the more powerful sensor nodes are able to perform special tasks. These are only a few examples of different characteristics and properties of WSNs.

This wide area of application scenarios induces also different security requirements for the respective scenario. For example, the security requirements for a WSN deployed for wildlife monitoring may not be as high as for a WSN used in health-care. Security certainly is also important for applications such as military or police networks, emergency response, or safety-critical business operations. For

example, after a natural disaster like a tornado, hurricane, flood, or earthquake, WSNs could be deployed for real-time safety feedback to assist the rescue teams.

## 2.3 Characteristics of WSN

A WSN is composed of a large number of low-cost sensor nodes (**SNs**) and one or several sink nodes or Base Stations (**BS**) [SACO06]. SNs are typically small wireless devices with limited computational power, radio transmission range, storage size, battery power, as well as attached sensing devices. Sink nodes are distinguishing devices with powerful computation capacity, large memory size, high power energy and long communication range so as to collect data from SNs. Sink nodes act as the gateway between WSNs and the end user. Due to a lack of infrastructure, SNs need to cooperate with each other so as to maintain link and routing information. Each SN not only acts as a host, but also as a router for data forwarding.

If the network consists of all the same kind of sensor nodes, it is called *homogeneous*. Some applications require different kinds of nodes. In this case the sensor network is called *heterogeneous*. A heterogenous network makes also sense, if some nodes must perform more tasks and services than others, i.e. these nodes take a special role in the network. These specialized nodes are called cluster-heads. Typical tasks of a cluster-head is to store information about nodes in its cluster group, e.g. what type of sensor data this group can provide, to collect periodically or in occurrence of certain events all sensor data and generate collective report, and as well as to route foreign messages through the inter-cluster network. In most security architectures, the cluster-head manages also security keys used to encrypt and decrypt messages.

Generally, a WSN has the following characteristics [SACO06]:

- *Ad Hoc Deployment*: SNs are deployed randomly and hence they do not fit into a conventional topology. Once deployed, they usually do not require any human intervention. Hence, the networks should be self reconfigurable and the setup and maintenance of the network should be entirely autonomous.
- *Dynamic Network Topology*: SNs may run out of energy (die) or new nodes may be added to the network. Hence, the network connectivity changes with time, resulting in dynamically changing network topology.
- *Energy Constrained Operation*: SNs are usually battery powered and deployed in an unattended area. The exhausted batteries cannot be replaced after the deployment. Hence, any protocol designed should take the energy

as a major constraint. WSNs are also constrained in terms of computational complexity and storage.

- *Infrastructure-less:* WSNs are primarily infrastructure-less. There is no central authority to monitor SNs.
- *Shared Bandwidth:* The radio channel in a WSN is typically broadcasted in nature and is shared by all the nodes within its direct transmission range. So, a malicious node could easily obtain access to the data being transmitted in the network.
- *Large scale of deployment:* A WSN is a large-scale network, in which thousands of sensors are randomly deployed to monitor surrounding environment or track a particular object.
- *Unattended Operation:* WSNs are usually deployed in a hostile environment, and operating in unattended mode. The exhausted or damaged nodes cannot be recharged or replaced after deployment.



## 3 Related Work

Research in wireless sensor networks has reached a high popularity in the past years. Countless projects exist elaborating different aspects of algorithms, optimizations, features, and applications. The research areas are mainly concentrated on energy efficient communication, code mobility, localization, operating systems, and security. There is no research work which combines all of these attributes together in one system architecture. This thesis is the first approach to develop an innovative system for wireless sensor networks that is energy-efficient and offers features for security, localization and mobility. This chapter surveys projects from each of these related areas.

### 3.1 Energy Efficient Communication

Regarding MAC protocols for wireless communication there are two different approaches: TDMA (Time Division Multiple Access) and contention-based protocols. In TDMA each node has its own time slot within a frame where only itself can send or receive messages depending on the protocol. The message exchange is handled on the basis of schedules which repeat periodically. In contention-based protocols there are no time restrictions for message transfer. Each node can access concurrently the medium at any time. The protocol tries to avoid collisions and when collisions appear it resolves them.

The most common contention-based protocol is defined in the IEEE 802.11 standard. Although not designed for wireless sensor networks it is the most common protocol in wireless communication. The only energy-saving feature of IEEE 802.11 is the avoidance of collisions. Some wireless sensor protocols are direct descendants of IEEE 802.11.

The Sensor-MAC (S-MAC) protocol [YHE02] is one of them. S-MAC is a slot-based protocol where each sensor node has alternating sleep and awake phases. The network is divided into clusters and all members of a cluster are awake or asleep at the same time. During sleep phase, the node turns off its radio, and sets a timer to awake itself later. All cluster nodes exchange schedules in an initial phase. A node becomes a *follower* if it receives a schedule from one of its neighbors. A *synchronizer* is a node that didn't get any foreign schedules

and decides to build its own schedule. This ensures that within a cluster only one schedule is used, i.e. the schedule of the first node, the synchronizer. If a node receives more than one different schedules it follows all of them. Such nodes are members of multiple clusters and have a higher energy consumption, because they have to be awake for each schedule. Within an awake phase all cluster nodes contend with each other for medium access. The contention mechanism of S-MAC is the same as that in IEEE 802.11, i.e., using RTS (Request To Send) and CTS (Clear To Send) packets. The medium is assigned to the node who first sends out the RTS, and the receiver replies with a CTS packet. S-MAC needs a strict timer synchronization in order to achieve correct functionality. Periodic synchronization among neighboring nodes is performed to correct their clock drift. An extension of S-MAC by *adaptive listening* is described in [YHE04]. If a node A notices an ongoing communication of node B whom it wants to send a message, it sleeps the time until B is ready.

A modification of S-MAC called Timeout-MAC or T-MAC is introduced in [vDL03]. In S-MAC all nodes need to be awake in the contention phase even if they have nothing to send or receive. T-MAC uses a specific timer  $T_A$  to shorten the awake phase if the node does not need to communicate. Obviously the  $T_A$  is smaller than the contention phase, thus the energy consumption is reduced. But if the timer  $T_A$  is chosen too small the node sleeps early missing possible message requests of other nodes. This *early sleeping problem* could even lead to unfairness. In further extensions of T-MAC this problem is solved by *future request to send (FRTS)* messages, but this increases again the energy consumption. Nevertheless T-MAC gains better energy results compared to S-MAC.

The Etiquette protocol described in [GI04] is a TDMA protocol and doesn't use synchronized awake cycles like in S-MAC and T-MAC. The idea is that each node announces its own *office hours* where it is ready to communicate. These time slots are established in the initial phase in the way, that neighboring nodes do not have overlapping slots. Thus, the collision risk is minimized, but at the same time the number of slots is increased. This has the negative effect of increasing latency. On the other hand Etiquette has a very good energy balance.

Token ring [IEE85] can be classified as a combination of TDMA and contention-based. Each node has its own time slot (token holding time) where it has to manage the communication with multiple contending nodes. The *Wireless Token Ring Protocol (WTRP)* [ELSV04] comes close to our approach proposed in chapter 4. WTRP was developed for mobile ad-hoc networks. All nodes build a single ring in the initial phase. The aim of WTRP is to maximize the throughput and minimize the latency without restraining the mobility. Energy efficiency is not considered in WTRP because the nodes are mobile devices with strong energy resources like Laptops or PDAs. A mapping of WTRP on wireless sensor networks is  $E^2$ WTRP that is described in [DLWW04].  $E^2$ WTRP aims to en-

hance the energy balance by dynamic adaptation of the token holding time. An active node can send more messages if the token holding time is increased. The frequency of token hand-over is decreased at the same time that reflect in lower energy consumption.

## 3.2 Code Mobility in WSN

Although there was an initial excitement about the idea of mobile agents in the late nineties, there are only a few projects that exist concerning agent systems on wireless sensor networks. The most famous project in this area is *Agilla* [FRL05] [FRL06].

Agilla is a mobile agent middleware for wireless sensor networks. It offers a special extension for *TinyOS*, an open source operating system for embedded systems. The aim of Agilla is to create mobile agents and to allow them to spread out their code and current status over a wireless sensor network. Through this local control, the mobile agents obtain more flexibility to find an optimal position for application specific tasks. Based on the multitasking ability of TinyOS, Agilla allows to run several agents on a single sensor node. The number of possible agents differs depending on the local memory capacity. Each agent is autonomous but shares various resources of the middleware with other agents, i.e. a *neighbor list* and a *tuple space*. The neighbor list contains the addresses of neighboring nodes. Mobile agents use this list for planning their next migration. The tuple space offers a decoupled communication capability among agents. It forms a shared memory architecture, where addressing is performed over defined field names instead of memory addresses. A tuple describes a typified data object. All tuples exist even if the inserting agent does not exist any more. Another agent could later reuse the data in the remaining tuple. In this manner the tuple decouples the sending agent from the receiving agent. Neither knowledge about the location nor the existence of the communication partner is required. For the implementation of agents Agilla uses a stack-based assembler language. This approach is the main disadvantage of Agilla, since even the programming of simple applications is really hard and needs a lot of code lines. Complex applications are nearly impossible. First step to enhance this was to provide a higher programming language which is automatically transformed in Agilla code. But still only simple command sequences are possible. Another weakness of Agilla lies in the realization of the tuple space. In a wireless environment it is obvious that inconsistencies can arise. Due to connection failures data could not be reached or several tuples with the same value could exist. A *global tuple space* would be a solution but would contradict to energy and resource requirements of wireless sensor networks. Agilla was established and tested on the sensor boards



*Mica2* and *MicaZ*, that have more extensive hardware resources than the sensor boards used in this work.

Another project that uses code mobility but not in the manner of mobile agents is *Contiki* [DGV04] [DFEV06]. It is an open source operating system for an 8-bit controller. Contiki is implemented in the language C and is already ported to several microcontroller. It is not an agent system but it offers the possibility to dynamically re-programm sensor nodes during runtime using mobile code approach. The code itself can not trigger the migration instead it is pushed by the underlying middleware. Contiki was designed for embedded systems with extremely low memory capacity. A typical Contiki configuration uses only 2 kilobytes of RAM and 40 kilobytes of ROM. The operating system consists of a simple event-triggered kernel, where applications can be loaded and unloaded dynamically at runtime. The processes of Contiki uses the so called *protothreads*, which provide optional preemptive multitasking. Inter-process communication is performed using message passing through events. Sensor nodes can communicate over a reduced TCP/IP stack called *uIP*. Contiki uses the standardized *ELF*-format [ELF95] for the dynamic binding and loading of code.

### 3.3 Location Estimation and Tracking

The *Active Badge* system [WHFG92] was an inspiration for several following projects. The goal of the Active Badge project was to easily locate persons in public buildings like hospitals. An application is to forward incoming phone calls to the current room phone near to the person's location. The active badges are devices worn and used to identify the person by sending out an infrared signal every 100 milliseconds. The use of common IR technology holds the production costs low. Active badges have a range of 6 meters and can run on battery almost for one year. Additional energy saving approaches can achieve a fourfold increase in service time. A network of sensors attached in each room receive the signals sent by these badges. Each sensor network is able to contain as many as 128 sensors which are connected to a workstation over a serial port. The workstations themselves are connected to a master that gathers and controls all sensor data. The Active Badge system can locate persons or objects in a room-wide range, but the resolution is very low and not sufficient. Another weak point is the high installation cost since all of the controlled area needs to be wired up, and extensions are hardly possible.

The *RADAR* project is a location tracking system based on wireless LAN [BP00]. RADAR is established in an area of  $980m^2$  with more than 50 rooms. Three base stations are used to cover the whole building, where the coverage of stations partially overlap. A laptop with a WLAN adaptor works as a mobile device

for locating and tracking. The laptop sends multiple UDP packets to the base stations that calculate signal strength and signal-to-noise ratio for each packet. It first builds a reference model with measurements consisting of 70 points in the building with data for each direction (north, west, south, east). RADAR stores at least 20 values of signal strength for each combination of location and direction. Additionally, it calculates the means, standard deviation, and median for each position. The accuracy of RADAR is similar to the Active Badge system. It is only possible to locate people in rooms. But the installation cost is lower, since most buildings provide WLAN access infrastructures. The calculation of reference data is the main disadvantage of RADAR. Each change in the room structure requires an update of the reference model.

Another project that uses WLAN for location tracking is described in [LBR<sup>+</sup>02]. This approach is similar to RADAR but uses a Bayesian interference algorithm for statical evaluation based on a specific model of state and observation spaces. The mean deviation could be improved compared to RADAR. Nevertheless this approach is also highly dependant on the room structure. Even small changes lead to large number of re-calculations of the reference model.

[HHS<sup>+</sup>02] presents another approach for location and identification of objects based on ultrasound. Each person or object carries a device called *bat* that sends periodically an ultrasonic signal. Receivers of this signal are *ultrasonic receiver units* which are attached to fixed positions on the ceiling. These units are interconnected to a daisy-chained network. Using base stations, the ID of a corresponding bat, which needs to be localized, can be sent over a wireless connection. The bat responds with the ultrasonic signal. Using different arriving times at different receiver units, the location of the bat can be calculated. This project shows that ultrasound provides high precision for location. On the other hand ultrasound is interference-prone. Other signals can easily jam the ultrasonic signal. Since also in this project the installation cost is very high, it is difficult to extend the system infrastructure.

Cricket [SBGP04] is another ultrasonic based location system. In this approach the device carried by the person determines itself the location. This ensures the privacy of the person. *Beacons* attached to the ceiling periodically send a radio and ultrasonic signal. Using multiple signals from different beacons the personal device calculates the current position. In further work, Cricket was extended to provide a tracking of moving objects. An *outlier rejection* component is used to eliminate measurement failures by deleting extremal values. Another component is the *least square solver* which has the task of minimizing squared mean failures. Current states are stored by an *extended Kalman-filter* that can even predict future states. The installation cost of Cricket is lower than other projects, but the interference problem of ultrasonic remains.

The technology of active RFID (*Radio Frequency Identifier*) tags is used in [NLLP04]. The aim of the LANDMARC project is to build a cheap location system that does not need sight contact and is insensitive to environment changes. LANDMARC uses RFID readers with a range of 150 feet. The range could be extended with specific antennas to 1000 feet. Additionally, the readers have an interface for wireless ethernet that allows flexible positioning. Each reader has eight reading ranges which can be changed incrementally. The reader can read out up to 500 tags in 7.5 seconds within each range. In a test scenario, four readers are attached in a large room. Since there is no possibility to get the signal strength, the readers have to scan all eight ranges sequentially. LANDMARC sends detected tags over wireless connection to a central computer. Using reference measurements, it calculates the location of the tags. The results show that LANDMARC is insensitive for changes in the environment, like persons in the area. The main disadvantage of LANDMARC is the sequential scan of all reading ranges that takes nearly one minute for each turn. Also the readers are relatively expensive, which affects the installation cost.

### 3.4 Security in WSN

Research in this area focuses basically on three security aspects: key management, secure routing, and verification of sensor data. This section describes related approaches that are relevant for each aspect and can be used in wireless sensor networks.

Critical factor in key management is secure and efficient distribution of keys to sensor nodes. Because of limited resources in sensor networks usually symmetric keys are used. Symmetric encryption requires that both communicating nodes know the same secret key.

**Key Management:** [CPS04] presents secure key distribution techniques for sensor networks. In particular, two approaches are described: single network-wide key and pair-wise shared key. The simplest method of key distribution is to preload a single network-wide key onto all nodes before deployment. Storage cost is minimal because each node has to store only one key. There is no communication needed to distribute the key. Unfortunately this approach provides sufficient security only if all nodes are protected against physical force. But this usually does not apply to low cost sensor nodes. An alternative method would be to distribute securely the secret key in the initial phase and after that delete all relevant information on nodes. However, new nodes can not join the network after this phase. The pair-wise shared key approach requires that every node in the sensor network shares a unique symmetric key with every other node. Hence,

in a network of  $n$  nodes there are a total of  $\binom{n}{2}$  unique keys, whereas each node has to store  $n - 1$  keys. The storage cost is proportional to the total number of nodes in the network. Therefore, the pairwise key scheme does not scale well for large sensor networks.

In [PST<sup>+</sup>02] a security protocol for sensor networks called *SPINS* was presented for hierarchical sensor networks with one or more trustworthy base stations. SPINS consists of two parts: a secure network encryption protocol (SNEP) and authenticated broadcasts ( $\mu$ TESLA). SNEP provides the security properties confidentiality, authenticity, integrity, and timeliness. Each sensor node  $u$  receives on a secure channel an individual, symmetric master key,  $K_u$  which is only known by the base station and the node. Using this master key the sensor node is able to generate all keys  $K_u^i$  by performing a one-way function  $F$ :  $K_u^i = F_{K_u}(i)$ . Storage cost for keys remains low, because each node has to store a fixed amount of keys independent of the network size. No additional communication between base station and node is needed, because each one can generate the keys independently. The disadvantage of SNEP is that secure communication can be built only between a base station and nodes, and it is not possible to protect the communication in or between clusters. The second part of SPINS is  $\mu$ TESLA that provides sending of authenticated broadcasts. For symmetric encryption, sender and receiver must share the same secret. Consequently, a compromised receiver is able to act as a designated sender by transferring forged messages to all receivers. In order to ensure authenticated broadcasts, an asymmetry between sender and receiver is needed.  $\mu$ TESLA uses delayed disclosure of symmetric keys for generating an asymmetry between sender and receiver. This approach requires weak time synchronization of sender and receiver in order to achieve time shifted key disclosure. Storage cost increases because each node has to buffer packets which it can only verify after receiving the key in the future time-slots. Also, this causes new possibilities for DoS-attacks. An attacker can force a buffer overflow by sending planned broadcasts. Furthermore  $\mu$ TESLA leads to scalability problems, which are described in [LN04].

An important aspect for sensor networks is that different communication patterns exist requiring different security steps. [ZSJ03] suggests an adjusted key distribution for different security requirements. For this reason, four different kinds of keys are used. The individual key is similar to the master key of SPINS. The second kind of key is a pair-wise shared key, which is generated in the initial phase for each known neighbor. Furthermore, the nodes have a cluster key for secure communication between cluster members. The last key is the group key that is used for secure broadcasting. This approach provides more flexibility but contains a security risk during the initial key distribution phase.

**Secure Routing:** Compromised sensor nodes can influence a sensor network,

especially by manipulating of routing information. In order to minimize the impact, [DHM06] suggests intrusion tolerant routing in wireless sensor networks called *INSENS*. The goal is to provide a working network even if parts of it are infiltrated. *INSENS* contains two phases: route discovery and data forwarding. In the first phase, the base station sends out a broadcast to build routes to each node. After receiving the route request, the nodes send a list of all known neighbors back to the base station. For the last step, the base station generates several disjoint paths to each node and sends this routing information back to all network members. Based on this routing table, the nodes can forward data to the base station (data forwarding phase). *INSENS* prevents the network against most outsider attacks and even insider attacks remain locally. But the dependance on the base station suggests a single point of failure. Furthermore, the route discovery phase is extremely energy inefficient.

Another approach for secure routing called *ARRIVE* is presented in [KLP02]. The routing algorithm tries to send packets over different paths based on probability. Also, nodes in *ARRIVE* listen passively to communication among neighbors. In case of detected failures, other nodes can forward the packet on behalf of its neighbor. *ARRIVE* works with smaller routing tables, but the chosen path is not always optimal. Furthermore, *ARRIVE* does not provide authenticated broadcasts, that provides a mechanism for manipulation of routing information.

**Verification of Sensor Data:** Each individual sensor node is potential target for attackers. Using compromised nodes, an attacker can directly influence the sensor network by infiltrating false reports of network sensor data. This kind of attack is called fabrication report attack. These fake reports can reach the base station, if they remain undetected, where they can trigger off false alarms. Also this causes high consumption of bandwidth and energy. En-route filtering attempts to verify reports on the way from sending node to the base station. The goal is to detect and discard false reports earlier. [ZSJN04] describes an interleaved hop-by-hop authentication scheme for filtering of injected false data. This algorithm recognizes infiltrated reports by a deterministic process, as long as no more than  $t$  nodes are compromised. The sensors build clusters with a minimum of  $t + 1$  nodes, where each group chooses a cluster head  $C$ . Only the cluster heads can send collected events in the form of reports to the base station. These reports include additionally to the actual event  $t + 1$  independent confirmations of the respective parties, in order to verify the authenticity of the report. Each intermediate node checks the report on the way to the base station (*En-route filtering*). Unfortunately, this approach uses single path routing to the base station providing several security risks. A statistical en-route filtering is presented in [YLLZ05] that enhances the approach above by using probabilistic algorithms. Each node chooses randomly a number of keys from a partition of a global key pool. Because of the probabilistic distribution of keys, any node can

---

verify with a certain probability a report before it is forwarded. In this manner, [YLLZ05] supports multi-path routing. But in both approaches, an attacker can create any report, once it has compromised at least  $t$  nodes. [YYY<sup>+</sup>05] attempts to solve this problem by binding keys to the location of nodes. The sensor area is divided into cells of width  $c$ , whereas each cell contains several keys. The nodes receive a location-bound key for each *sensing cell*. This approach bounds reports to their original location.



## 4 Energy-Efficient Communication

Many sensor nodes depend on battery energy that should preferably last a long time. Even if a sensor node has an alternative energy source, like in EnOcean [EnO07], the energy is so small that energy saving remains essential. In most applications battery replacements are not feasible because of the great number of sensor nodes or the unknown or inaccessible location. Imagine a scenario where sensor nodes are spread out from an airplane.

A sensor node consumes most of the energy in its active mode. The energy cost rises enormously if the node uses radio communication. The energy consumption of the microprocessor Texas Instruments MSP430F149, which is used in several sensor boards, is  $1.6 \mu\text{A}$  in sleep mode and rises to  $280 \mu\text{A}$  in active mode at 1 MHz. In [YHE02] the energy rate between active:receive:send is determined as 1:1.05:1.4, i.e. the energy consumption of sending a message outweighs other tasks. It is plausible that the energy consumption can be highly decreased by establishment of sleep intervals and avoidance of unnecessary packet sending. Using sleep intervals can lead to a contrary effect, i.e. the data exchange is reduced to a shorter time, which can cause higher packet collisions and increase energy consumption.

The *Energy Saving Token Ring Protocol (ESTR)* proposed in this thesis maximizes the lifetime of the overall wireless sensor network by using sleep intervals [BUB08a]. The nodes in ESTR are connected to a ring where a token is passed along. The node that holds the token is allowed to communicate to other nodes and can control the communication if there are several nodes contending with each other. Other nodes that do not need to communicate sleep in the meantime. ESTR avoids idle-listening and overhearing, i.e. the receiving of foreign messages where the node is not the recipient. Unfortunately sleep phases decrease the throughput and increase the communication latency. In order to remain flexible ESTR offers the possibility to set the maximum size of a ring and allows to build multiple interconnected rings. By changing this parameter you can balance between energy-saving on the one side and throughput and latency on the other side. The token in ESTR has a special field for energy information. Each node fills this field with its own energy level before it sends the token to its next neigh-



bor. The successive node can adapt his sleep period based on the energy level of its predecessor. This mechanism offers a dynamic self-optimization where the ring balances the overall energy and increases its lifetime.

## 4.1 The ESTR Protocol

The *Energy Saving Token Ring Protocol* is based on the idea of the wireless token ring protocol, but chooses a different communication mechanism and introduces energy-efficient techniques. ESTR assumes that sensor nodes can not change their location if attached to the environment. New nodes can join the sensor network or some sensors can fail due to energy loss, but this is not assumed to be in an ad hoc manner like in mobile ad hoc communication networks. Using sleep phases is still the best approach to save energy. ESTR introduces sleep periods for sensor nodes which does not need to send or receive messages. Thus nodes spend less time in active mode and even receive less foreign messages reducing the overhearing problem.

Usually in token rings only the node which holds the token is allowed to send messages. In ESTR we have reversed this communication scheme by allowing the token holding node to *receive* messages, because the reception consumes less energy than the sending.

Using sleep periods leads to a problem because nodes don't necessarily know if the neighbor is awake or not. This problem rises if the two nodes are in different neighboring rings. Since the token is sent in a message the node has to be sure that its neighbor receives the message. To ensure full token ring functionality, ESTR offers solutions for token loss, multiple tokens, and node failures that are described below.

### 4.1.1 Network Structure and Configuration

The sensor network in ESTR can consist of one large ring or multiple smaller rings that are interconnected. Each structure has its advantages and disadvantages. One ring built by all nodes comes close to the original Token Ring, but this is not always possible and depends on the location of nodes. Imagine a sensor node with only one neighbor where the role of predecessor and successor is given to one single node. As long as all nodes can reach each other, we can first build a tree structure to determine the Eulerian cycle as a ring structure. Since we use radio transmission, the edges in the tree are undirected. That means that the father node and son node in the tree can reach each other, as seen in Figure 4.1. The Eulerian cycle is obtained by visiting each node of the tree in accordance

to a depth-first search and putting the edges in visited order to a path together. The path starts and ends at the root and reaches all nodes of the tree. This leads to some nodes that are visited more often than others, if they have several children, but it creates a circular structure. Another possibility is to modify the algorithm of WTRP so that the predecessor and successor nodes for a new node can be the same. This allows the integration of nodes in the ring, which can only reach one node. In both algorithms one node needs to be declared as *root* which initializes the ring structure. The modified WTRP solution costs less energy, since no additional messages need to be sent to organize the ring structure. The Eulerian ring algorithm on the other hand brings more structure in the ring, which can be used for further improvement.

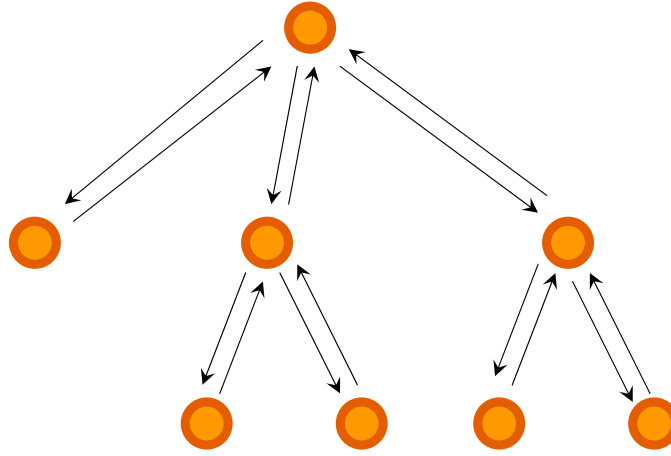


Figure 4.1: Eulerian cycle

The process of building multiple interconnected rings except for some minor changes is similar to the process of building one large ring. First, each node in the network has to be a potential root of a ring, i.e. after a random time the node decides itself to create a ring if it didn't receive an invitation from an existent ring in its vicinity (see Figure 4.2). Interconnected rings require advanced collision handling, since foreign messages can cross ring borders. The number of collisions also increases in the initialization phase, as several rings are built in parallel causing more message traffic. On the other hand, building smaller rings reduces initialization time.

In ESTR we can control the number of rings in the network by setting a maximum size of ring members for each node. Within the initialization phase the root node sends invitation messages until the maximum size of members is reached. In order to finish this initial process there is an initialization timer. After it expires, each ring remains with the obtained number of members, even if the maximum is not reached. Controlling the size has a direct effect on the latency and throughput

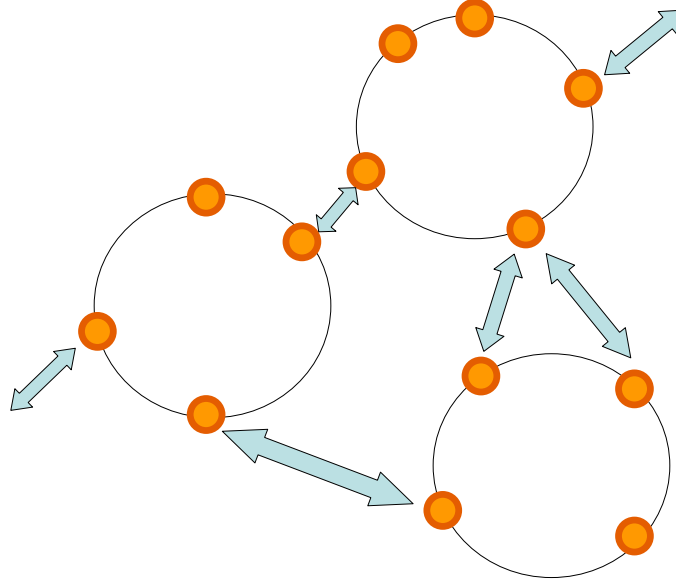


Figure 4.2: Multiple interconnected rings

of the network. It is application dependent which results in a ring size.

#### 4.1.2 Node Communication

As mentioned above in ESTR only the node that holds the token receives messages from neighboring nodes. After expiration of the token holding time ( $THT$ ) the current node  $A$  has to hand over the token to its next ring neighbor  $B$ . In order to be sure that  $B$  is awake and able to receive the token,  $A$  waits for a  $M_{idle}$  message from  $B$ . If  $A$  receives this messages in the time of  $T_{waitMax}$  it forwards the token to  $B$ . The time  $T_{waitMax}$  is two times longer than the maximum sleep period of the nodes  $T_{sleepMax}$  to ensure enough time for token hand-over.

If node  $B$  doesn't receive the token after sending  $M_{idle}$ , it repeats the  $M_{idle}$  message. If the token is still missing after  $T_{waitMax}$ ,  $B$  stops sending  $M_{idle}$  assuming that its predecessor failed. On the other side node  $A$  will only go asleep when  $B$  stops to send the awake message  $M_{idle}$  ensuring that the token is received correctly. Through this secure transmission a token can only be lost if the token holder fails or if the token holder cannot reach its neighbor in  $T_{waitMax}$ . In both cases the process of token recovery is the same. If the token could not be transferred in the designated time, either the token holder or the token expecting node will induce the dissolution and then the reorganization of the ring by sending respective messages. If the network consists of multiple smaller rings the

reorganization is executed with only few messages.

Nevertheless delay or failure in communication can lead to the case that more than one token can exist in the ring. In order to solve this problem each token has two numbers: the *sequence number (SN)* and the *generation sequence number (GSN)*. *SN* is used to count the number of passed nodes. *GSN* can only be increased by the root and counts the number of rounds. A node deletes all tokens with a *SN* or *GSN* smaller than the last sent token. The root which can only change *GSN*, deletes all multiple tokens.

Assuming that there is no node failure and the token is transferred correctly from *A* to *B*, the communication can go on. In this case, node *A* sends its data to *B*, since *B* as the new token holder is now allowed to receive data. *B* acknowledges each received data packet by sending an *ACK* message. Figure 4.3 describes the message exchange.

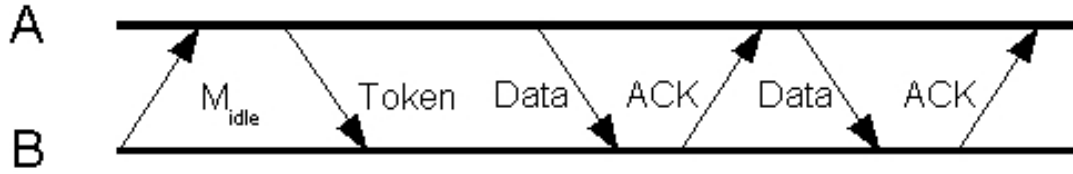


Figure 4.3: Token hand-over and message exchange

Other nodes from neighbor rings that desire to send data to *B*, have to wait until *A* has no more packets to transfer. This initiates the contention-based phase of ESTR. After the last packet of *A* the other nodes set a random timer before they send a request to *B*. Node *B* decides if it has enough time before *THT* expires and sends a permission to the first node. This process reflects the RTS/CTS-approach of IEEE 802.11 that is also used in S-MAC and T-MAC.

The nodes cyclically change their state during the lifetime in order to react correctly to incoming messages. All possible states and transitions are described in Figure 4.4. After the initial phase all nodes are in the *sleep*-state except for the root node which generates the token at first. The root is in the state *sendToken*. As soon as its successor awakes, it changes to the *waitToken*. After the state *sendData* a node can either go to sleep or wait for another node in a neighboring ring if it wants to send data to it. In some cases, a node could miss its sleep phase waiting for other nodes and changes directly to the *waitToken* state. Figure 4.5 shows all sections of the awake phase.

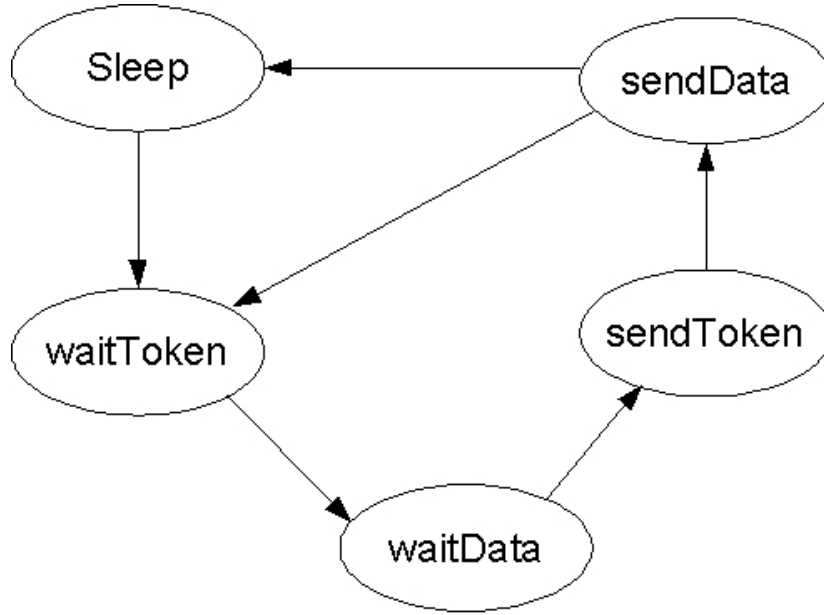


Figure 4.4: State chart of the ESTR-MAC-level

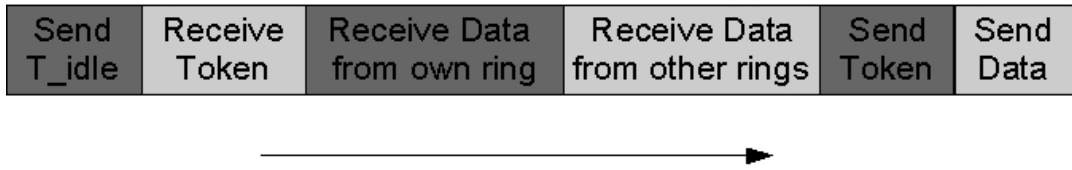


Figure 4.5: Sections of the awake phase

### 4.1.3 Self-optimizing by Dynamic Sleeping

The length of the sleep phase depends on the number of nodes in a ring. The more nodes are in the ring the longer a single node can sleep. But this affects also the message transfer time because the path through the ring becomes longer. A node that has to send many of messages to neighboring rings cannot use its sleep periods so often and will fail faster due to energy loss. In order to balance the energy consumption in the ring, ESTR offers a dynamic self-optimizing approach where nodes can change their sleep duration. ESTR uses the token to transfer energy information from node to node.

The token holder  $A$  uses the energy field of the token to inform the next neighbor  $B$  about its energy level.  $B$  increases its sleep duration by a constant factor  $c$  if the remaining energy of node  $A$  is factor  $c$  higher than the energy level of  $B$ .

The effect of this adaptation is seen as  $A$  forwards the token again to  $B$ , i.e. in the next cycle. This time  $A$  has to wait longer for  $B$  who sleeps longer. If the energy level between  $A$  and  $B$  is balanced,  $B$  sets back its sleep duration. Since node  $A$  has now a higher energy consumption, it can also adapt dynamically its sleep duration. Over the time the energy consumption of the entire ring becomes dynamically balanced, increasing the overall lifetime of the sensor network.

## 4.2 Evaluation

In order to evaluate the performance of ESTR and to compare it with related approaches, we used the network simulator NS-2 from University of Berkeley, California [The07]. The protocol stacks of IEEE 802.11 and S-MAC were already available in NS-2. We implemented the protocols T-MAC, Etiquette, and WTRP according to information given by the authors.

### 4.2.1 Average Energy Consumption

The average energy consumption is an essential factor for the lifetime of the wireless sensor network. A high energy consumption leads to faster node losses. We measured the energy consumption of ESTR in comparison to the other protocols, increasing the number of nodes in the sensor network. Figure 4.6 visualizes the energy consumption of a wireless sensor network with 20 nodes. The chart shows clearly that IEEE 802.11 does not have energy saving possibilities except for CSMA/CD. It loses the most energy as compared to other protocols in this scenario with 20 nodes.

S-MAC as well as T-MAC have a higher energy consumption compared to 20 nodes because the nodes have to send and receive more messages, e.g. in the synchronization phase. ESTR and Etiquette offer nearly the same energy efficiency. In ESTR, the network consists of multiple rings, hence, the initialization phase does not need more time. Regarding the overall results, the average energy consumption increased compared to the smaller network. As a last step we increased the number of nodes to 500. Figure 4.8 shows that the task of building the rings takes disproportionately more time.

WTRP cannot manage to build the ring within the defined time of 300 seconds. After expiration of this time the nodes start to reorder. In a second step we set up the timeout for building the ring to 400 seconds. Even this was not enough to manage the initial process, and WTRP loses a large amount of energy after the timeout. The reason for this is that the remaining nodes try to build their own ring after 300 seconds. The invitation to join the ring is sent from node to node

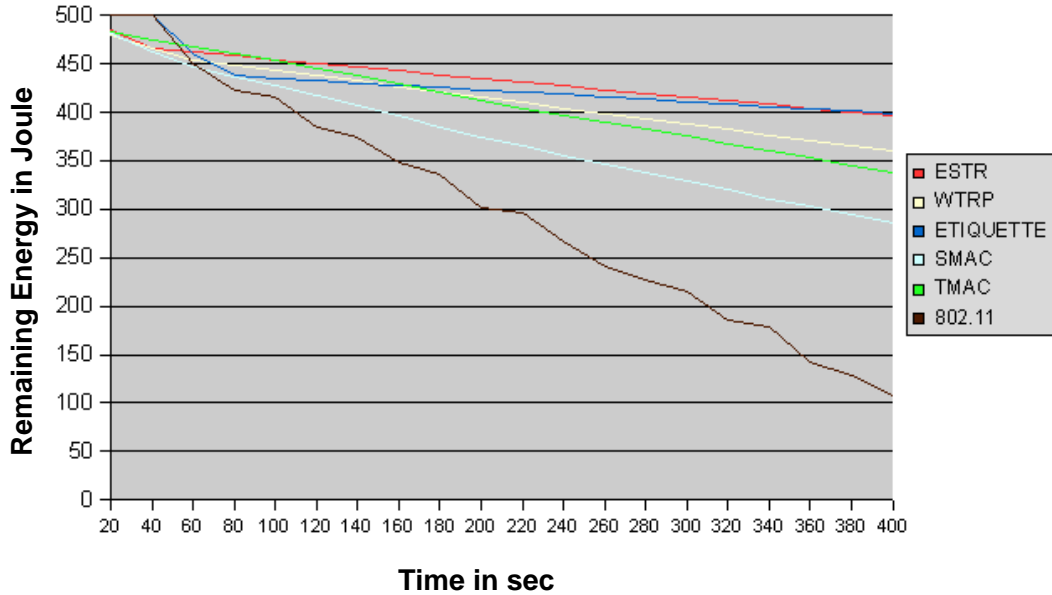


Figure 4.6: Energy efficiency with 20 nodes

It is important to note the high energy loss of Etiquette and ESTR at the beginning. After some time both protocols decrease linearly as S-MAC and T-MAC, but gaining the best energy results at the end. ESTR as well as WTRP build only one ring since the maximum number of nodes per ring is set to 30 nodes. The initialization costs for smaller rings are still acceptable, as clearly seen at WTRP that has similar results as ESTR and Etiquette. S-MAC is factor of two better than the 802.11 protocol due to the sleep phases. T-MAC with its timeout mechanism is even better than S-MAC.

What we can also see here is that even in smaller networks TDMA- and token-based approaches are more efficient than contention-based protocols. The difference rises if the number of nodes becomes higher. Figure 4.7 shows the energy consumption in a networks with 100 nodes. This time WTRP needs more time for building the token ring.

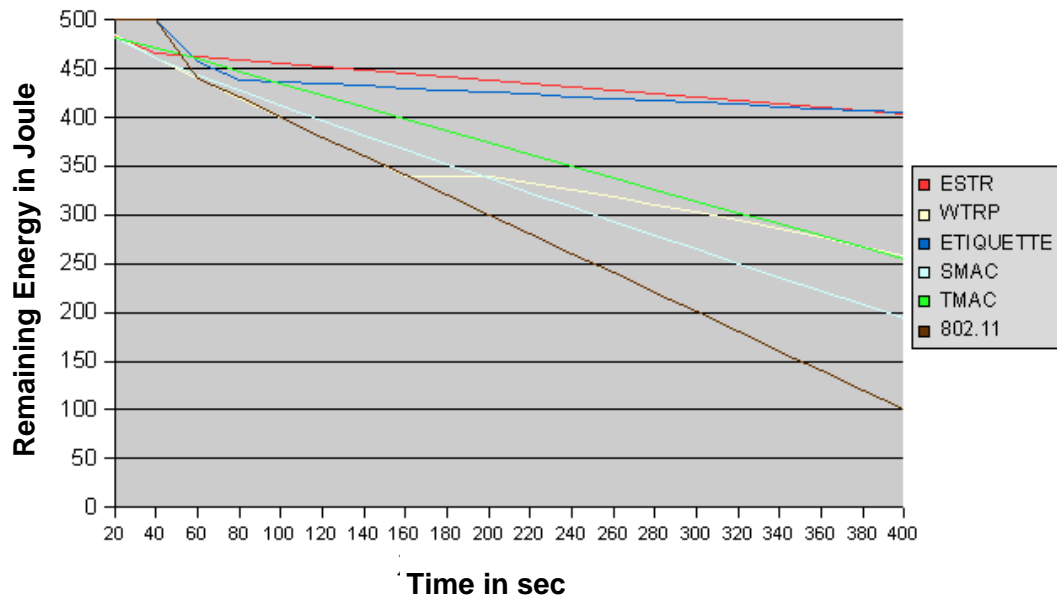


Figure 4.7: Energy efficiency with 100 nodes

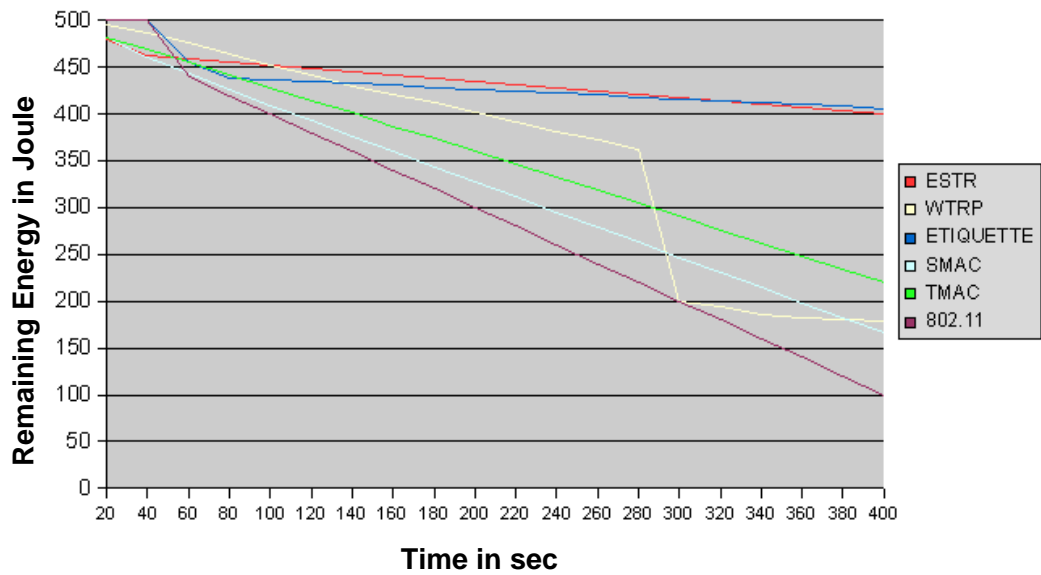


Figure 4.8: Energy efficiency with 500 nodes



in WTRP. This becomes a problem if there are too many nodes. At 100 nodes WTRP needed 150 seconds to reach each node, at 500 nodes it can not reach most of them in 300 seconds. Another problem is the calculation of the node sleep time. In order to compute the sleep phase which depends on the number of actual nodes, the token has to be passed through the ring at the end of the initialization. Assuming an awake time of 0.25 seconds and 500 nodes, the token should take 62.5 seconds for one circulation to conclude the building of the ring.

This is also the reason why ESTR as well as Etiquette are more energy efficient. ESTR also suffers from having too many nodes. The packet collisions increase retransmissions, which consume more energy. Also it could be that some roots cannot get new members since they are surrounded by existing rings. But the great advantage of ESTR is that these nodes can still communicate as rings with only one member having their own sleep periods. Thus, the initialization phase does not need to be extended in ESTR using the same time like in smaller networks.

In the beginning phase, ESTR seems to consume more energy as WTRP. The reason is that in ESTR multiple roots try to build rings in parallel. In WTRP there is only one root that sends invitations to other nodes. But ESTR quickly compensates this negative energy balance after the initialization.

In the previous evaluations we did not use the self-optimizing mechanism of ESTR using dynamic sleeping. In order to examine the impact of this feature for energy consumption, especially the balance of energy between nodes, we ran some measurements using a network with only 30 nodes.

If a node loses faster energy than its neighbors, then the loss of this node will impact the network greatly. The remaining nodes will have to take over the tasks of the lost node. Thus, we consider the node with maximum energy as well as the node with minimal energy to draw conclusions about the lifetime of the overall network. Etiquette, ESTR and T-MAC can offer the best results (Figure 4.9). The proportion between awake and sleep phases have not changed, but ESTR has now the possibility to balance the energy in the sensor network by dynamically adapting the sleep phase.

Since the lines of S-MAC and IEEE 802.11 overlap in the chart, the values of S-MAC can not be recognized directly. The timeout functionality of T-MAC ensures that the energy of the node with the least energy does not decrease rapidly compared to S-MAC. In ESTR one can see improvements because of dynamic sleep adaptation.

The next chart (Figure 4.10) shows the energy quotient.  $E_{Min}$  is the energy minimum at a point in time and  $E_{Max}$  is the energy maximum at the same time. Thus, the energy quotient is  $E_{Quot} = E_{Min}/E_{Max}$ . With this value we can see the

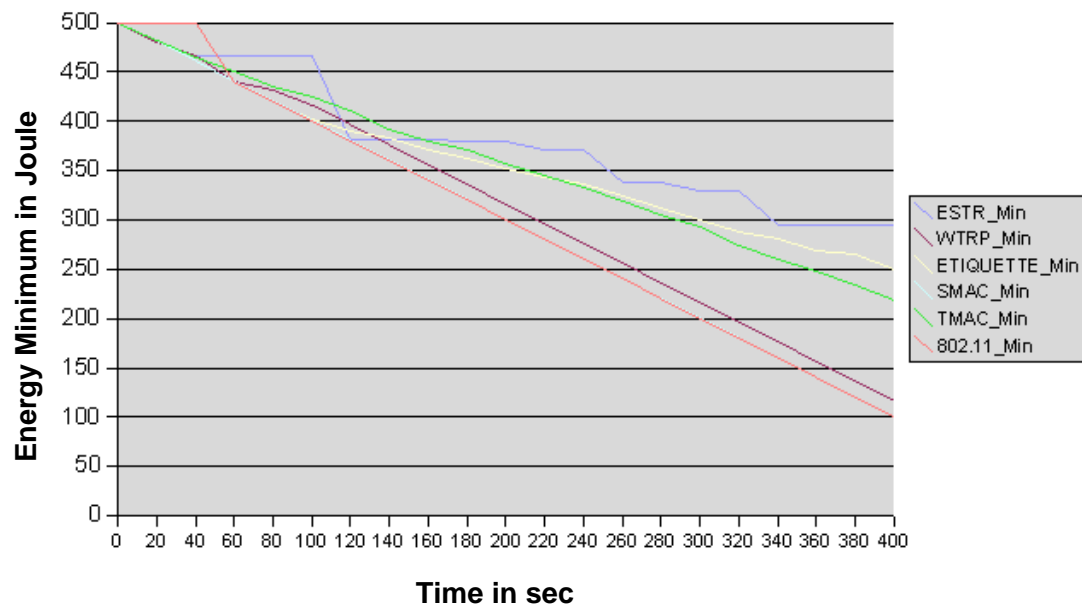


Figure 4.9: Energy minimum in the sensor network

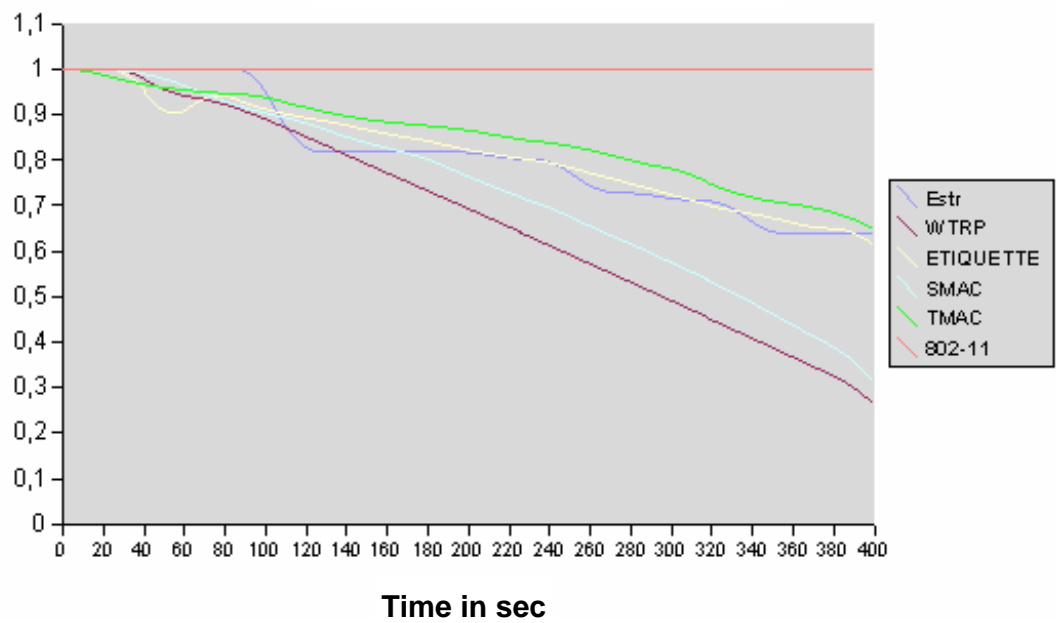


Figure 4.10: Energy quotient

relative distance between minimum and maximum of energy. In IEEE 802.11 the minimum and maximum are always equal. The nodes have no sleep periods and the number of sent and received packets is for each node the same. S-MAC and WTRP have the worst results. It is assumed that in S-MAC the synchronizer has the lowest energy since it has to manage the initialization. Its the same with the root node in WTRP. The Etiquette protocol does not have a designated node that leads to a better energy quotient. ESTR can compensate the difference at the initial phase through dynamic sleeping. In some phases the energy is balanced, and the minimum and maximum are equal. In T-MAC the energy quotient is decreasing constantly which means that the minimum and maximum lose the same amount of energy.

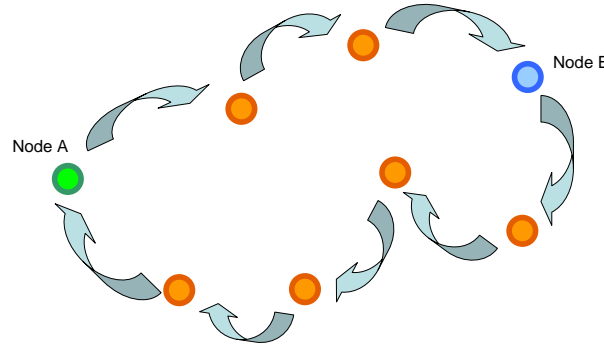


Figure 4.11: Ping-pong between two nodes

#### 4.2.2 Communication Latency

Even if the energy consumption is the leading point of ESTR, the latency is noteworthy. In applications with time-critical events, latency becomes a main focus. In order to measure the delay we chose a *ping-pong* communication where a message is sent from node *A* to *B* and immediately back again to *A* (Figure 4.11).

Especially in ring structures where messages between two physical neighbors often have to pass all nodes of the ring this approach is more significant than the latency of single packets. In most of the network structures the latency of a ping-pong message is determined by halving the transmission time, since the node can reach each other directly.

In our latency evaluation we used a single ring with 30 nodes. Thus, the results of ESTR also represent WTRP regarding the latency. We compared our results

with S-MAC and IEEE 802.11. First we chose two nodes that were physical as far away as possible. The 802.11 protocol doesn't have sleep phases, thus it has the least latency as seen in figure 4.12. The first chart has a large scale, where the value of 802.11 is not visible. The second chart uses a smaller scale to estimate the time of 802.11. In ESTR the message is sent in one or two cycles through the ring. Each cycle consists of awake and sleep phases as mentioned above. In S-MAC the slots for sending are short. The message has to pass multiple intermediate nodes. At each station it has to wait another sleep phase to be forwarded. This results in large delays, if the nodes are far away from each other.

If the physical distance between the nodes is small, e.g. neighboring nodes, the results change. Figure 4.13 shows the ping-pong times using close-by nodes. In ESTR nothing changes because the message still has to pass through the whole ring, even if the nodes are physical neighbors. S-MAC as well as IEEE 802.11 have one third shorter times. In S-MAC there are still sleep phases that result in transfer delays.

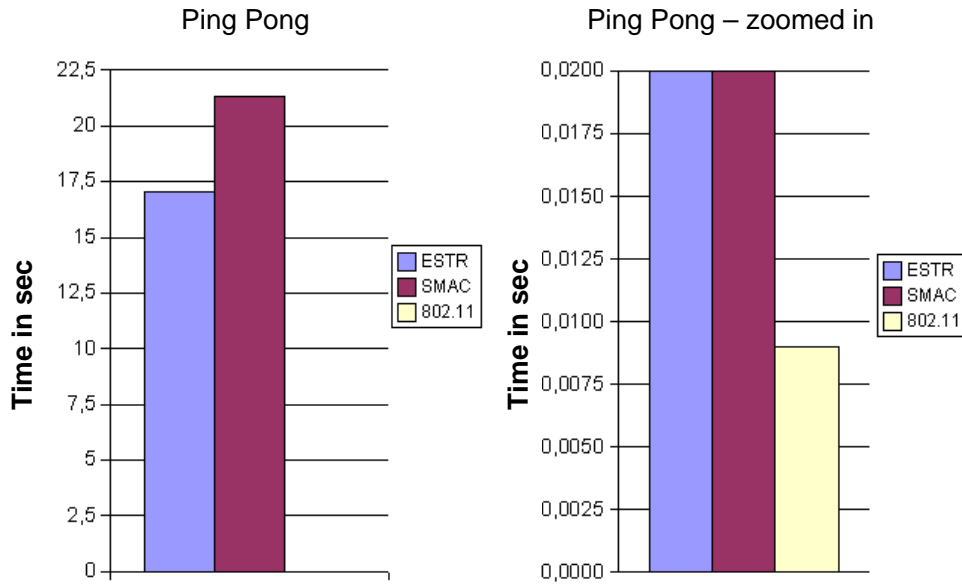


Figure 4.12: Time for a ping-pong between far nodes

### 4.2.3 Real-life Tests

Beside the NS2 simulations we implemented ESTR on sensor boards to evaluate its usability in real-life scenarios. The used hardware was the *Embedded Sensor Board (ESB 430)* from the FU Berlin [Sca07]. This sensor board has the micro-

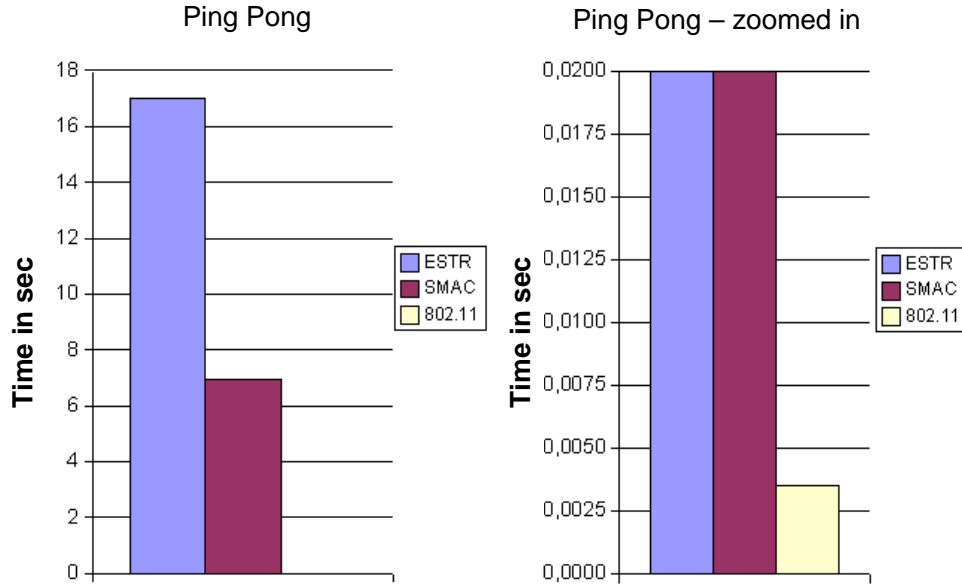


Figure 4.13: Time for a ping-pong between close nodes

controller TI MSP430, a radio transceiver, and several sensors on board. The MSP430 is a low-power microcontroller offering four different sleep modes. Thus the sensor board ESB430 is appropriate to test energy-efficient protocols.

We implemented two scenarios: one ring with three sensor nodes and two rings with three nodes per ring. One node was always connected over the serial port to a PC where we could get visual outputs in order to observe the communication. Figure 4.14 describes the one ring scenario whereas Figure 4.15 shows the two interconnected rings.

To establish the desired ring structure, we started the nodes successively. The first node that was attached to the PC became the root. In the first scenario, the other nodes sent a message to each other every 96 seconds, after joining the ring. Since every message passed the root, we could follow the ongoing communication. In the two ring scenario we set the maximum member size to three. Successively starting of the nodes should result in two different rings interconnected through one or more nodes. To test the ring-to-ring communication we sent messages from the root node to all other nodes which answered with an ACK packet.

The firmware of the ESB430 has an interface where you can get the remaining energy of the battery. This led us to the idea also to measure the energy consumption of ESTR in real-life. We implemented the IEEE 802.11 protocol to compare the energy usage with ESTR. Since the firmware does not provide accurate energy values, battery energy is surprisingly lower at the beginning of the measurement

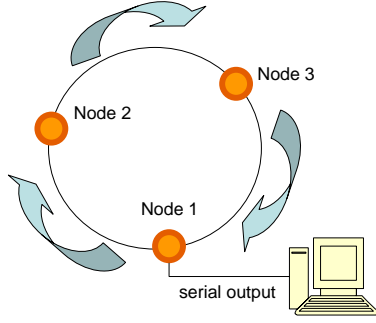


Figure 4.14: Real-life scenario with one ring

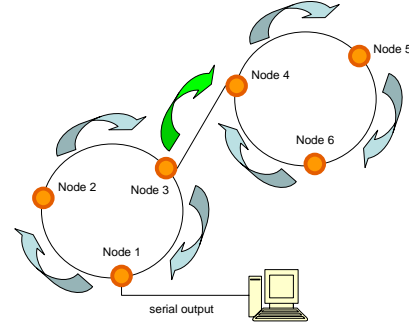


Figure 4.15: Real-life scenario with two interconnected rings

and fluctuates during the time (Figure 4.16). Nevertheless it is clearly seen that ESTR achieves a better energy balance as IEEE 802.11 although it is a small network with only three nodes. The main advantage of ESTR becomes apparent in larger networks.

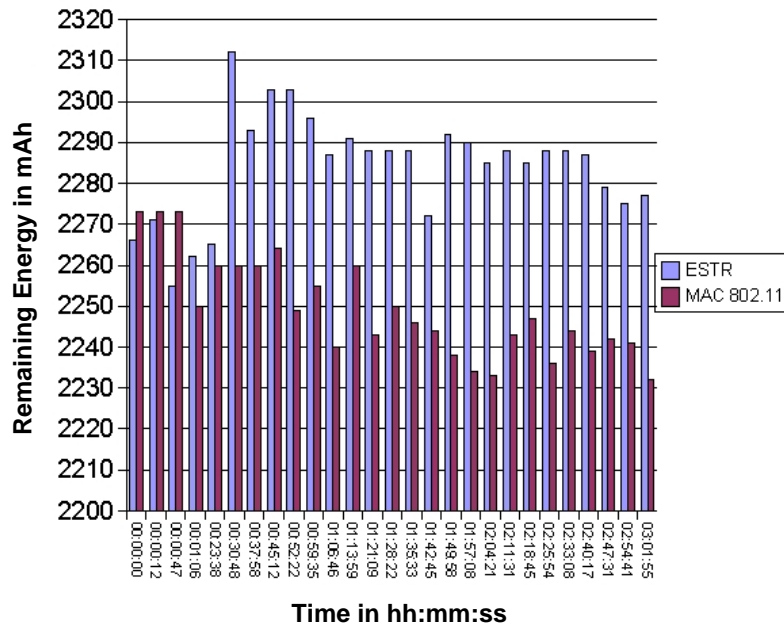


Figure 4.16: Energy consumption in real-life



## 5 Variable Ranges Protocol

The energy consumption of sensor nodes that send with maximum transmission power lies significantly higher than with reduced power. Decreasing transmission power results in exponentially decreased energy consumption. Additionally, the initial state of reduced transmission power of nodes ensures that complexity of network is low. With high signal strength nodes are confronted with frequent interferences and redundant paths within the network. Low signal strength means that number of neighboring nodes is less, i.e. probability for message collisions decreases.

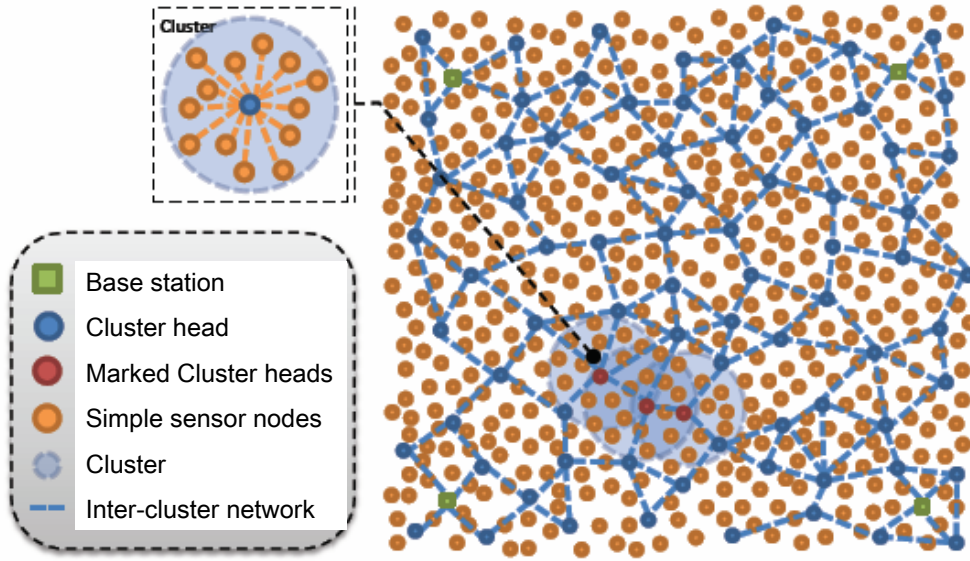


Figure 5.1: Basic sensor network architecture

Often, sensor networks form a hierarchical topology based on clusters, each containing a number of simple sensor nodes and one powerful node that acts as a cluster-head. Sensor nodes connect directly to the cluster-head, i.e. routing in clusters is not necessary. Regarding our topology approach in this thesis, we decided also to build on a cluster architecture. In our view of a sensor network, a node can be a member of several clusters at the same time. All cluster-heads



form together an inter-cluster network used for sending messages to base stations. Furthermore, it is assumed that sensor nodes do not change their position once they are attached to a location. The network works with multiple base stations in order to avoid single-point-of-failures (see Figure 5.1).

In first versions of our proposed system, clusters were built in the initial phase and remained unmodified for the whole lifetime of the network. Furthermore, all nodes used the same sending configuration, i.e. transmission power and range were set to maximum on each node. This chapter describes an extension of this approach with dynamic features. The new variable ranges (VR) protocol optimizes the communication range of each node [BKJS11]. Therefore, the VR protocol saves energy by adjusting and optimizing the signal strength to particular circumstances of the sensor network in order to extend the lifetime of nodes. This optimization results in more efficient usage of energy throughout the overall sensor network [Bag11].

Regarding the security architecture described in the next chapter, the number of neighbors is also an important parameter. Each cluster-head has to manage several keys with each other neighboring sensor node and cluster-head for securing the communication and ensuring authentication. More neighbors means higher management effort and more storage space, as well as more encryption and decryption processing. All of this results again in increased energy consumption. Therefore, it is essential that the security architecture works hand in hand with the underlying communication protocol.

## 5.1 Network Establishment

In the initial phase of the VR protocol, the nodes search for neighboring nodes starting with a minimum signal strength. For this reason, each node sends a *DiscoverNodes* message containing its own range parameter and ID. Then the node waits a certain time to get a response. The waiting time is also dynamic, i.e. the time is low, if the range is low and increases, if the range increases. The reason for this is that a node with a low range will reach less neighbors. Therefore, there is no need to wait a long time for a response. The nodes increase stepwise their transceiver power and send new discovery messages until a pre-configured number of nodes is found. A node which receives a *DiscoverNodes* message of an unknown node, extracts the range information of its seeking new neighbor. In the next step, the node compares the received range value with its own range. If the own range is lower, the node increases its range and sends a *DiscoverNodesReply* message back. Since the signal strength of the nodes would increase in this way until all nodes would settle at the range of the largest distance between two nodes, the VR protocol performs only a temporary range adjustment. This

means that nodes discard the adaptation after a certain time and return again to their previous values.

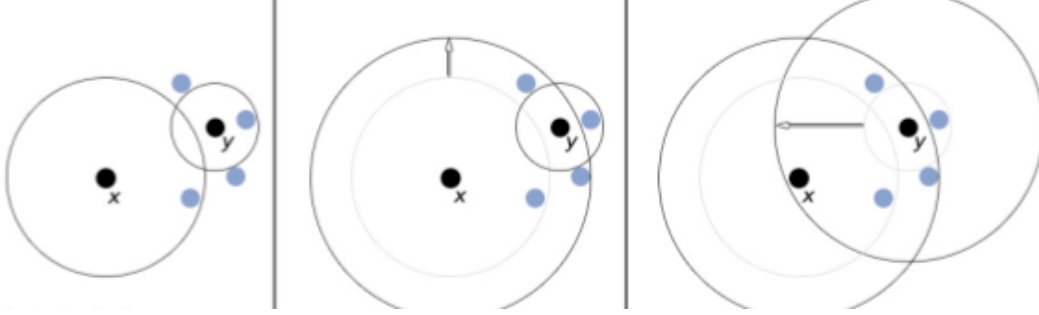


Figure 5.2: Range adjustment in VR protocol

Figure 5.2 illustrates this adjustment scheme. Assuming a maximum number of neighbors is set to three in this figure, it would be unfavorable for node  $y$  to use the range parameter of node  $x$ , since it can reach three neighbors with a much lower signal strength. For this reason,  $y$  will only increase range temporarily to answer node  $x$  and return to its previous range, in order to proceed with its own search. Cluster-heads find in this way a minimum number of other cluster-heads and as well as sensor nodes.

Actually, the aim of each cluster-head is to be reachable through the inter-cluster network by at least one base station. After the initial phase, each base station sends a broadcast through the new built network. This sink message is also important to generate new keys for further authentication. In the next chapter, this key generation is described in detail. Therefore, reachability of base stations is essential to establish the security architecture in the sensor network. If a cluster-head does not receive a broadcast message after a certain time, it starts a new search phase to find new cluster-heads. The cluster-head uses this time a different message *DiscoverNewNodes*. It first uses the current range, since there could be cluster-heads which are in range, but not discovered. Cluster-heads who receive such a message, adjust their signal strength to answer, but keep their new range value this time. If the node does not get any answers, it increases its range and repeats the procedure until it finds new connections. Figure 5.3 shows the *TryToConnect* phase. You can see on the right side of the figure, that after the initial search phase, several local cluster networks are established, but not all of them can reach a base station marked here as green squares at the four corners. On the left picture you can recognize that the connectivity is enhanced after the *TryToConnect* phase, but nevertheless there are still local clusters remaining unreachable by any base station. The reason is that nodes are deployed randomly. There is a small probability that some nodes cannot reach a base station, even

if transmission power is set to the maximum or due to message collisions in the initial phase.

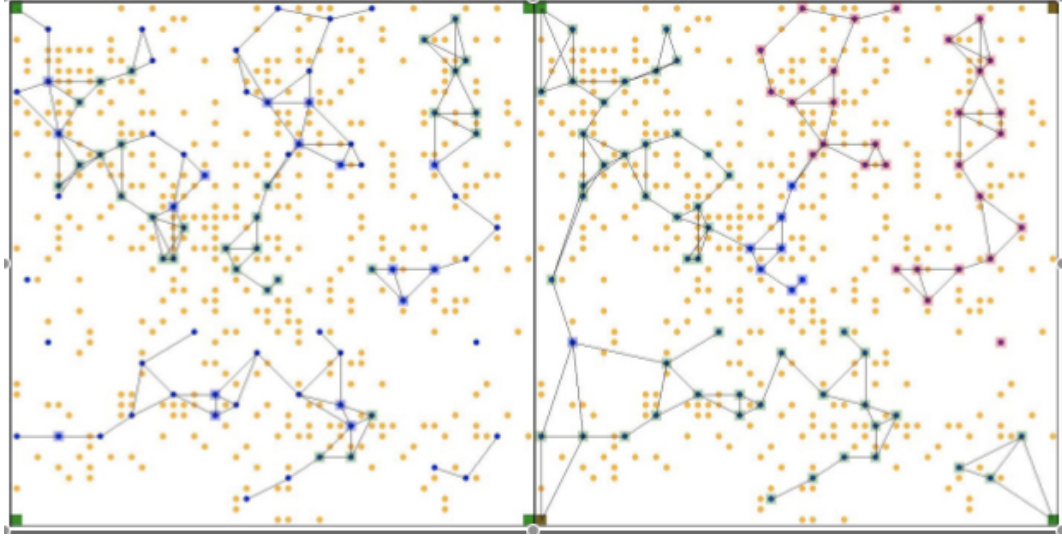


Figure 5.3: Initial phases of VR protocol: a) neighbor search b) TryToConnect

## 5.2 Probabilistic Routing in VR Protocol

Cluster-heads check periodically their neighborhood for node losses or new arriving nodes. During lifetime the VR protocol ensures that nodes can dynamically adapt to changes in their environment. This network adaptation goes hand in hand with security adjustments.

Additionally, routing information is updated by cluster-heads after each VR adjustment phase. Simple sensors do not need routing capability, because they exclusively communicate with the cluster-head. Routing is used only within the inter-cluster network established by cluster-heads. Our architecture uses probabilistic multi-path routing based on the level values to forward messages from cluster-heads on the way to the corresponding base station. Cluster-heads build up a trust matrix, where each transmission to its neighbors is recorded. Based on this trust information, cluster-heads calculate a probability value and write it into the packet header. This value is used to decide in which direction the packet has to be send. Each cluster-head modifies the probability value and sends the message over the most trustworthy route.

Furthermore, our architecture provides passive participation, i.e. sensor nodes listen to packet transmissions of their neighbors. If cluster-head  $u$  detects a

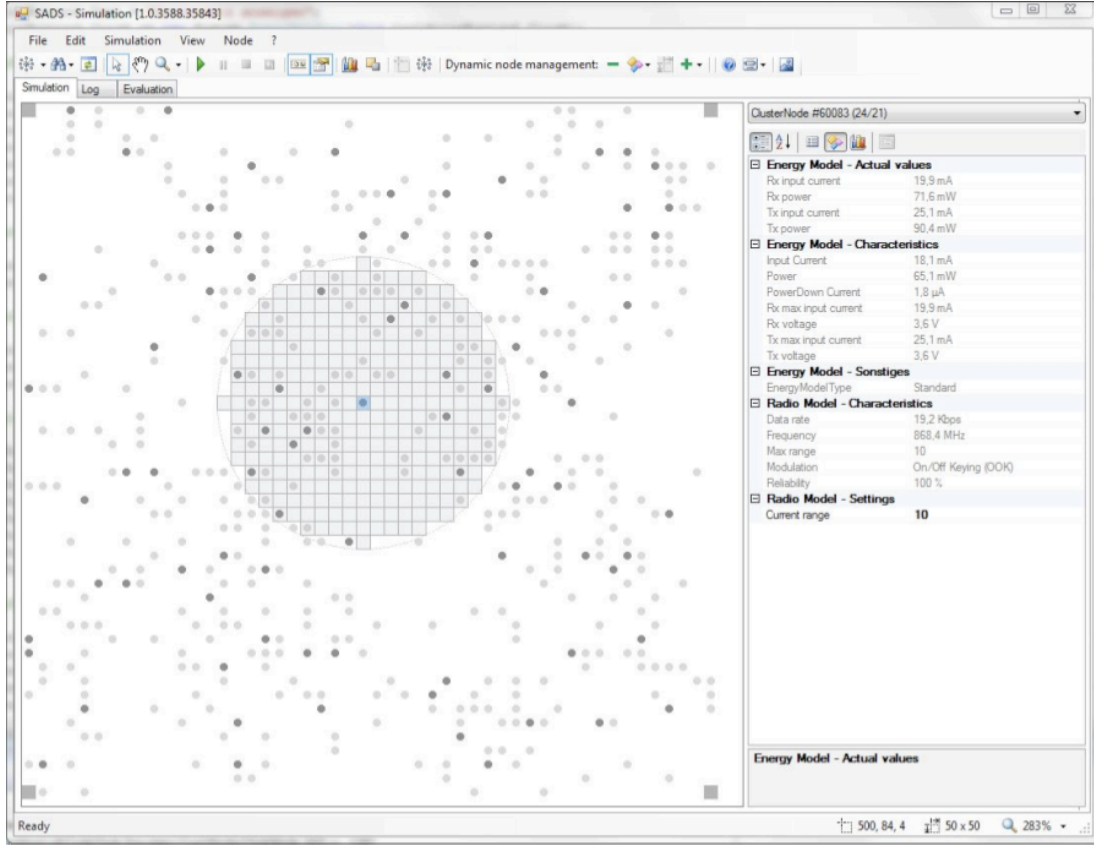


Figure 5.4: The Simulator GUI

packet addressed to its neighbor  $v$ , and recognizes that  $v$  is not forwarding the message,  $u$  takes responsibility with a certain (low) probability. Also, if  $u$  assumes that  $v$  forwards the message to a non-existent node,  $u$  takes care of transferring.

## 5.3 Evaluation

To evaluate the efficiency of our variable ranges security architecture we implemented a simulation tool where it is possible to establish different sizes of sensor networks. Figure 5.4 shows the GUI of the simulator. The simulation is divided into three phases: node distribution, initialization of network, and report sending. In the first phase, a predefined number of nodes is distributed randomly over a given area. Sensor nodes and as well as cluster-heads are deployed after setting for each a maximum transceiver range.

Basic parameters for the network are total number of nodes, initial node range,

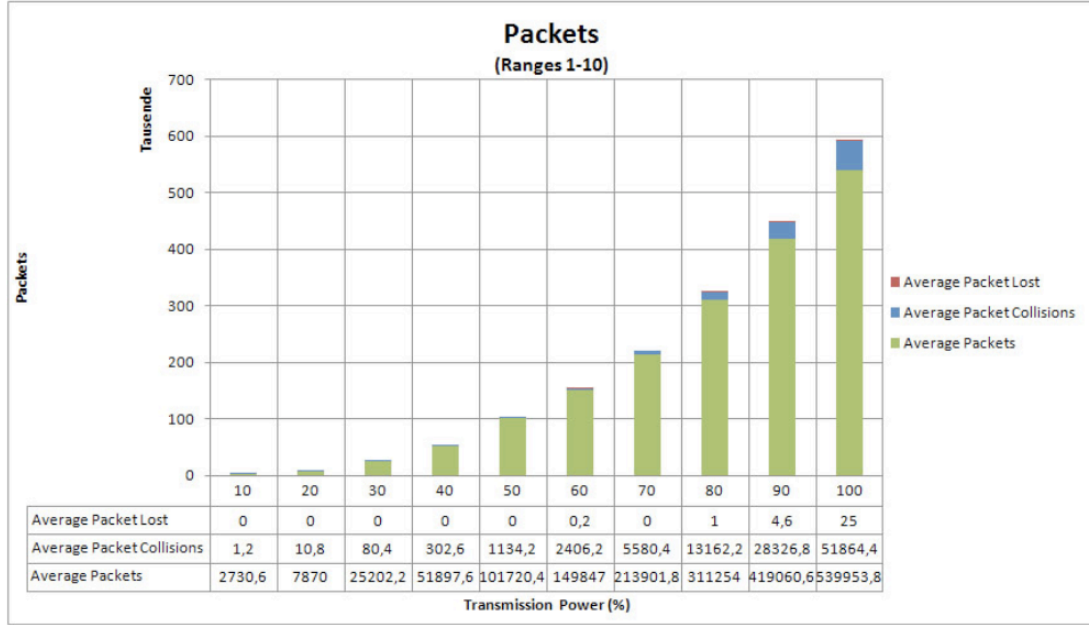


Figure 5.5: Traffic load in relation to signal strength

initial node energy, and network density. Type, range, and position of nodes can be changed easily using the simulator GUI. Furthermore, new nodes can be added or existing nodes can be deleted before the next phase of the simulation is started. Figure 5.4 shows a screenshot after the first phase. Dark circles are cluster-heads whereas light dots are simple sensor nodes. The squares at the corners represent again four base stations. In this case one cluster-head is selected and you can see the communication range of the current node.

The second phase initializes the network based on a communication protocol. We implemented three protocols that can be selected by the user in the beginning of this phase. These are the SMAC protocol, the energy saving token ring protocol (ESTR), and the variable ranges (VR) protocol. The user can change a set of parameters depending on the selected communication protocol, e.g. cluster size, timer settings, update periods. In this phase security and routing information is exchanged, too.

At the end of initialization, the network is established and nodes can start to exchange secured messages. This is simulated in the last phase by randomly generated reports that are sent to base stations. The user can halt the simulation at any time, in order to change parameters for nodes. For example, one can turn off a node to simulate a node loss. It is also possible to simulate a compromised node that sends false reports into the network.

Nodes consume energy for processing data, like encryption and decryption, and sending or receiving messages. For some communication protocols nodes can switch to a sleep mode, where energy consumption is minimal. Our simulator bases on an energy model that uses specifications of real sensor boards: ESB 430/1 and MSB-430 of Freie University Berlin [Sca07].

In a first evaluation we measured the number of messages sent in the initialization phase using the VR protocol. We performed several simulations where we modified the maximum range of nodes in order to get average number of sent packets, collisions, and lost packets. Figure 5.5 shows the results of a network with 500 nodes. Traffic load increase with higher range of single nodes, because nodes can reach more neighbors to exchange messages with.

The VR protocol can be initialized with several parameters. As mentioned in the previous section, the VR protocol continues to search for new neighbors by increasing the transmission range. Using the simulator the user can set the maximum number of neighboring cluster-head and sensor for each node in the network, e.g. setting the number to three would stop the search after finding three cluster-heads in the neighborhood. In some cases, this would lead to a low connectivity, since nodes which could not join a cluster group would be disclosed. Therefore, VR protocol offers a second optimization step that was described in the previous section (*TryToConnect*). Figure 8.6 shows number of sent packets, packet collisions, and lost packets using different configurations of VR protocol. The notation *Init VR 3-3-ExtCon* means that each cluster-head searches for new neighbors until 3 other cluster-heads and 3 sensor nodes are found and the *TryToConnect* mode is turned on. It is clearly seen that packet collision in VR protocol is very low and there are nearly no packet losses, since the communication range is very reduced. The less packets are sent, the less energy is consumed. In Figure 5.6 the configuration *Init VR 2-3 noExtCon* seems to be the optimal configuration.

But regarding the connectivity this is not the best choice. Figure 5.7 shows the connectivity for each configuration. It is clearly seen that the connectivity for the configuration *Init VR 2-3 noExtCon* is only %14.15, i.e. only a small number of cluster-heads can actually reach a base station. Turning on the *TryToConnect* modus brings only an increase of %10 in connectivity. Only after increasing the number of neighboring cluster-heads leads to reasonable results. Even without the second optimization phase, VR protocol can reach nearly %90 connectivity.

As mentioned in the previous section, the complexity of sensor network is much lower using VR protocol. On the left side of Figure 5.8 the network was established with maximum node range. It is clearly seen that in dense areas of the network, the number of different connections is rather high. In a second simulation, we used the VR protocol to establish the network. As seen on the right

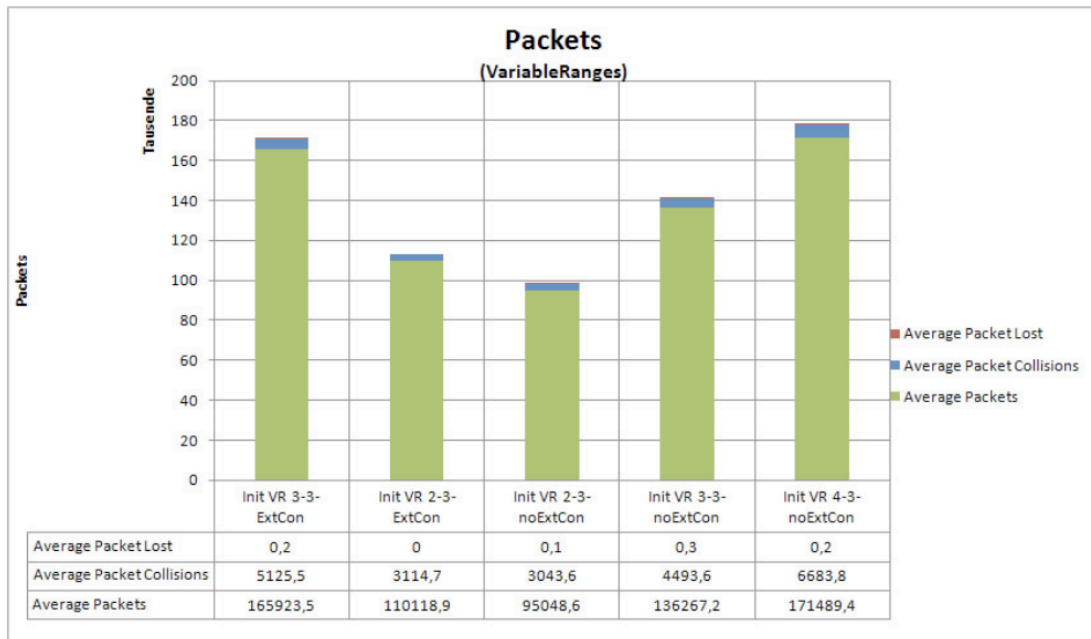


Figure 5.6: Traffic load for different configurations of VR protocol

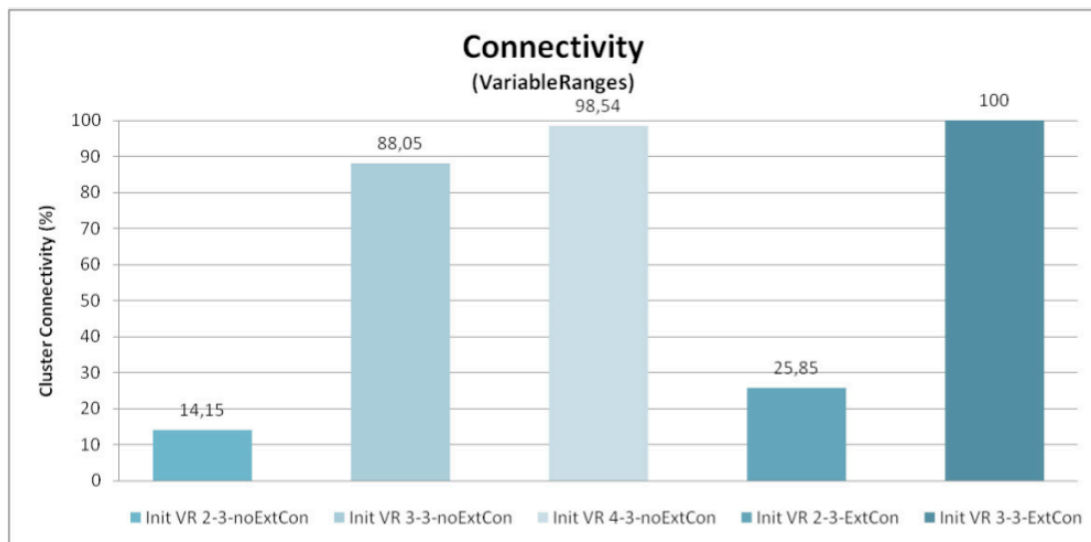


Figure 5.7: Connectivity for different configurations of VR protocol

part of Figure 5.8 using the VR protocol lowers the complexity of the network.

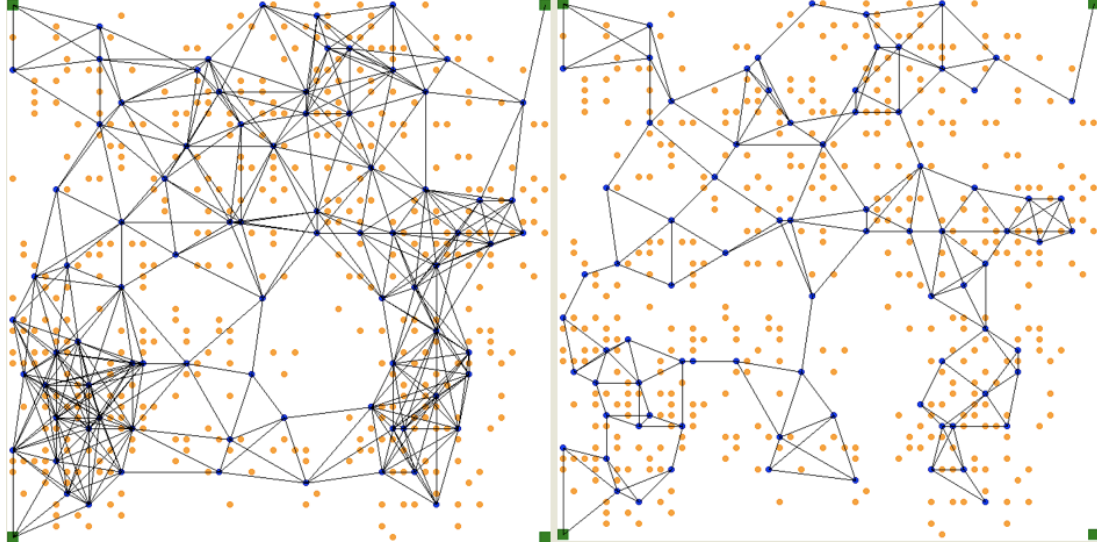


Figure 5.8: Network complexity without (left) and with VR protocol (right)

The optimal usage of signal strength in VR protocol shows its advantages also in energy consumption. Figure 5.9 illustrates the energy consumption for sending reports from a sensor node to the base station. Level represents here the distance between sending node and nearest base station, e.g. level 6 means that messages traverse six intermediate cluster-heads until they reach the base station.

We performed the energy measurement in four different networks with the same size. For the first three networks the maximum range parameter of each node was set to a fixed value, i.e. range 6 stands for 60 maximum signal strength. The last network used the VR protocol with at least three cluster-head and three sensor node neighbors and a further optimization step to increase the connectivity (*VR 3-3 ExtCon*). One can clearly see that the VR protocol has the best energy balance leading to a longer lifetime of the network.



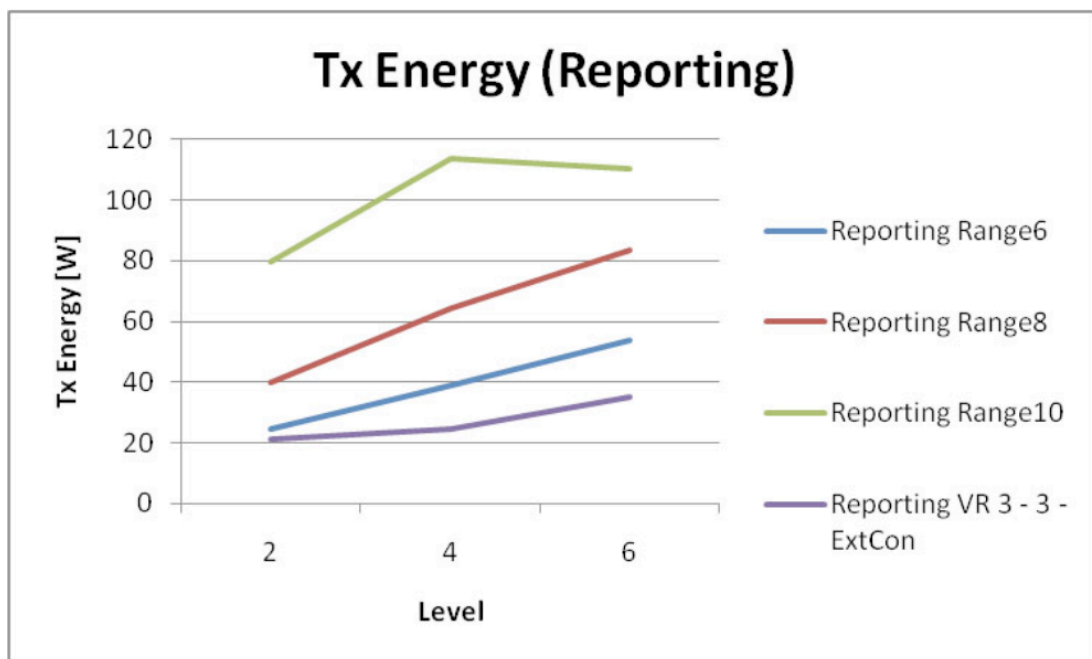


Figure 5.9: Energy consumption for reporting

## 6 Mobility in WSN

A research field that often integrates wireless sensor networks in their applications is *Ubiquitous Computing*. Ubiquitous systems build on the vision that great amounts of microchips and sensors will be integrated in everyday objects. The aim of microchips in this area is to support the users in an independent and invisible way instead of forcing them to adapt to the computerized world. Most of the devices will be attached to a fixed location whereas some will be mobile carried by persons. Service development on memory and energy limited devices is a complex task. An example for a service is a fire detection application on basis of temperature sensors that are spread out in an area. A program running on such a device is static and usually limited to a single task. If a new service needs to be performed each device has to be fundamentally reprogrammed. Regarding the large number of sensor nodes envisioned for future sensor network applications, this could lead to an intractable task. Running multiple services on one device is usually not possible because of memory and performance limitations.

Regarding the decentralized approach, the openness, and heterogeneity of wireless sensor networks, the paradigm of mobile agents presents itself for ubiquitous systems. The idea in this thesis is to bring a service on limited devices and sensors when it's needed and to switch services dynamically. Imagine a smart office building that has multiple microcontrollers and sensors integrated in walls, doors and office equipments, and users who have portable devices as well as microchips integrated in badges. In the initial state, the microcontrollers perform no services, but have the possibility to host mobile agents. The service comes in form of an agent to each device and leaves if it is no longer needed.

Agents in general can be defined as software units with certain autonomy. They perform services by order of a user or other agents. Mobile agents have an additional property: they can autonomously migrate, i.e. they can transfer program code, data and continuation pointer to a remote computer and resume with the program execution. Beside this physical mobility, mobile agents have the possibility to communicate with each other in order to exchange information.

This chapter elaborates the idea of mobile code in wireless sensor networks. Code mobility is provided by a mobile agent system for ubiquitous environments called *UbiMASS* [BWSU10], [BWUB09]. The mobile agents in UbiMASS can receive information from the real environment through appropriate interfaces. The light-

weight design of UbiMASS allows it to run on wireless sensor networks, that consist of several sensor nodes.

## 6.1 UbiMASS Architecture

Based on the requirements of ubiquitous agent systems, this section presents the ubiquitous mobile agent system for sensor networks. UbiMASS describes an agent middleware, which offers a fundamental basis, to develop a range of applications from the ubiquitous computing area.

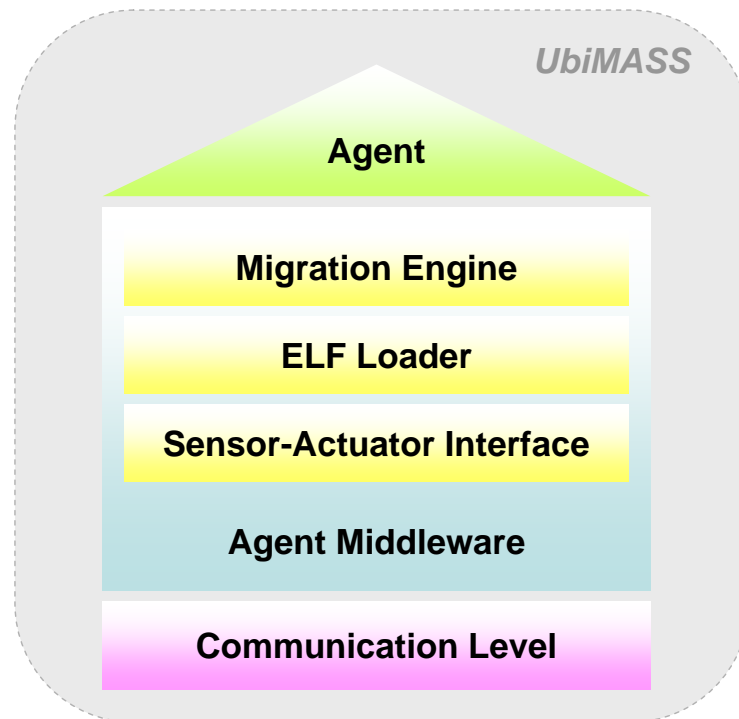


Figure 6.1: UbiMASS host architecture

The primary goal of UbiMASS is to offer mobile agents the opportunity to distribute on their own initiative the code, as well, existing data, and the current state over a wireless sensor network. The agents, using their own local control and flexibility, perform application specific tasks. It is not necessary to transfer data over an unreliable wireless connection because the agent itself has been moved to the data. This approach is an elegant and energy-efficient solution for distributed applications. Calculations proceed only at relevant locations sparing the entire communication network from useless traffic loads. UbiMASS consists

of multiple wireless connected agent hosts offering a platform for mobile agents. A UbiMASS host has a modular architecture with several components. Besides the management of the agent system, the hosts must run the agent and control its communication and migration.

Actually the agent middleware has three components: the ELF loader, the Migration Engine, and the Sensor-Actuator Interface. Figure 6.1 shows the UbiMASS host architecture.

### 6.1.1 UbiMASS ELF Loader

The agent system UbiMASS uses the standardized format ELF for dynamic loading of the agent. In the case of a agent migration the middleware calls the loading method of the ELF loader to begin the execution of the agent. In order to explain this operation it is necessary to describe at first the basics of the ELF standard.

ELF stands for Executable and Linkable Format, that is basically a standard for executable files. It is mostly used in UNIX systems where it replaces the old and unflexible *a.out*. ELF aims to rearrange the machine code in order to load it fast and efficiently into the memory for execution. The standard describes how commands have to be organized, interpreted and loaded. The representation of control data is, in contrast to other proprietary formats, always platform independent. [ELF95] distinguishes between three kinds of ELF files (object files):

- **Executable File:** This type of file comprises a program that is ready to be executed. All necessary information to create a new process is available. This process has access to the code and data within the corresponding file.
- **Relocatable File:** In this case the file does not contain an executable program, but only position independent code and data. These can be linked with other object files to produce an executable program or a dynamic library.
- **Shared Object File:** This file comprehends also code and data, that can be linked on two ways. On the one hand a new object file can be created using other relocatable or shared object files. On the other hand executable or shared object files can be used to produce a process image.

The object files are required for binding or linking of the program as well as for executing the program. Therefore, ELF defines different views of the same file: linking view and execution view (see Figure 6.2). The linking view constitutes the file as an alignment of *sections*. The execution view divides the file into *segments*. Each view has a table describing the several sections. Notice that sections and segments describe actually the same file only from different points of view. The

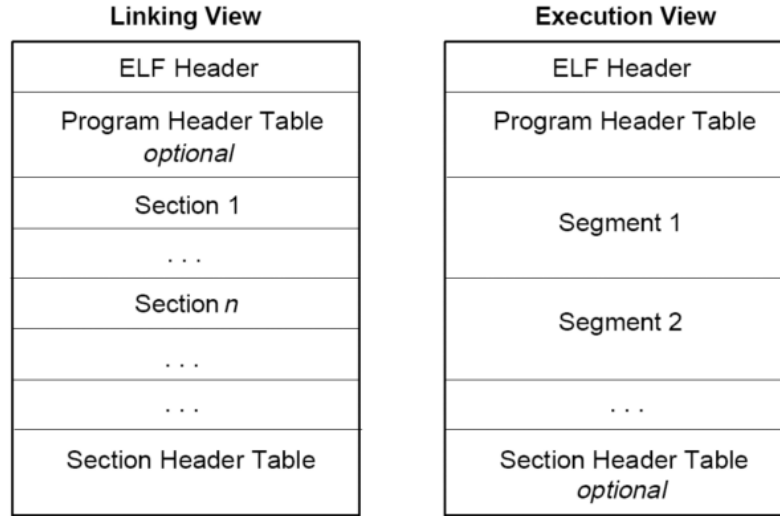


Figure 6.2: Different views of an ELF file

sections are used by compilers, assemblers, and linkers to arrange the file into several parts. In this manner the executable code is located in the section *.text*, all initialized variables in the section *.data*, and all not initialized variables in *.bss*. Additionally, there is a *string table* which contains all names of sections and symbols. Another essential section is the *symbol table*.

A symbol constitutes a kind of pointer with a name and a value. A name of a function or a variable can be associated with a physical address in this manner. The management of all symbols is done with the symbol table to allow an easy way of locating symbols. Besides index, names, and addresses of symbols the table also contains in the column *info* additional information about the visibility of the binding and the classification of symbols. The binding can be defined as *local*, *global* or *weak*. Local symbols are not visible outside of the object file similar to a local variable. Globally defined symbols are visible in all object files, i.e. they must be unique. Weak symbols actually behave like global ones, but have lower priority.

Actually an agent in UbiMASS is a relocatable ELF file. This file is transferred from one agent host to the next in case of a migration. In order to load the agent, the receiving agent host calls the function *elfloader\_load()* which performs the following steps:

- At first the header is checked to ensure that it is a correct and compatible ELF file.
- The next section header is designed to get the number and size of the

according entries. Using the string table the names of all other sections is determined.

- Now all sections can be parsed. The section *.text* contains the current program code of the ELF file. The section *.data* presents all initialized data, whereas *.bss* contains the non initialized data. for the relocation is stored in both sections *.rela.text* and *.rela.data*. Furthermore the symbol table and the string table are used to set the pointer on all the relevant sections and to store the section number for dissolving the final addresses.
- Two functions which are ported for the architecture of the sensor board ESB 430 are used to allocate memory. Sections *.data* and *.bss* are stored in the RAM, whereas section *.text* gets into the flash memory.
- Using the symbol table next step is to relocate in sections *.text*, *.bss*, and *.data*. This leads to the integration of the location independent code into its environment where it can be executed later.
- Finally the completely linked code is written into the allocated space. The loaded programm can now be started.

This mechanism offers the possibility to integrate easily new agents into the running agent system UbiMASS which ensures high flexibility and reliability.

### 6.1.2 Migration Engine

The Migration Engine of the UbiMASS agent middleware is responsible for complete and correct process of the agent migration. One of the key features is that the agent can decide on its own which current values of variables it needs to take to the destination host. In this manner, the agent can trigger an entry point in order to continue to work where it stopped at the previous host.

The process of the migration is shown in Figure 6.3 and described as follows:

1. Before starting the migration the agent can send current values of integer and string variables to the agent middleware using two functions *set\_init\_data(int id, int data)* and *set\_init\_data(int id, char data[])*. Because the agent itself will receive this list of tuples, there are no further specifications necessary. The agent knows how to handle the data on the destination node.
2. Using *start\_migration()* the agent initiates migration and lets the middleware to perform the required steps.
3. Agent middleware initiates the migration. Actually, it is a weak migration, because only the program code and data of the agent is transferred, but not

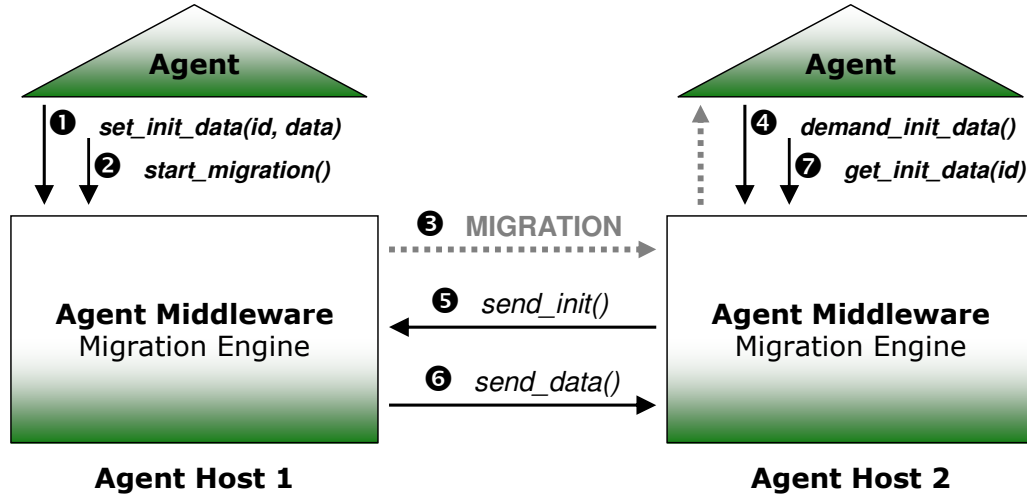


Figure 6.3: Agent migration in UbiMASS

its status. The agent starts from the beginning at each time. The agent and its memory address that is known from the initial loading is forwarded to the communication level. On the destination host, the agent is loaded and started using the ELF loader described in the previous section.

4. In order to receive the last values of its variables the agent calls at the beginning the middleware function *demand\_init\_data()*. This turns the migration into a strong migration, because the agent can now continue to work with its previous status.
5. The destination host requests the list of variables by sending an *INIT* to the source host.
6. The source host packs all tuples into a char array and sends it to the requesting host. Figure 6.4 gives an example of the char array.

i	32	3	s	4	t	e	s	t	...
---	----	---	---	---	---	---	---	---	-----

Figure 6.4: Example: sending of variable values

At the beginning of each variable the type is specified by an *i* for integer and *s* for a string. Because integer variables require 16 bits two chars are used to deliver one integer right-shifting the first bits by 8. The value in the example can be calculated by  $(3 \ll 8) + 32 = 768 + 32 = 800$ . After the string type, length of the string is indicated. After successful transfer of the

message, agent host can delete the agent process and deallocate memory space or agent can remain, if cloning of agents is desired.

7. After waiting for the variable value to receive the agent can access values using middleware functions *get\_init\_data(int id)* and *get\_init\_string(int id)*. Because the agent knows in which order it has put the variables, it also knows which variable matches its id.

The Migration Engine works as a fundamental component of the overall agent system. For this work our goal was to develop and to evaluate the basic functionality.

### 6.1.3 Sensor-Actuator Interface

Ubiquitous systems use context information to adapt to environmental changes. These changes can be detected using sensors. UbiMASS provides a sensor-actuator interface that is used by the agent system to access sensor and actuator information from the hardware. The access is strongly dependent on the used sensor board and its firmware. In the current version of UbiMASS the sensor-actuator interface is ported to the ESB430 sensor board [Sca07] that has multiple sensors and actuators on-board. Figure 6.5 gives an overview of current components of the UbiMASS interface. Porting to other devices is possible because

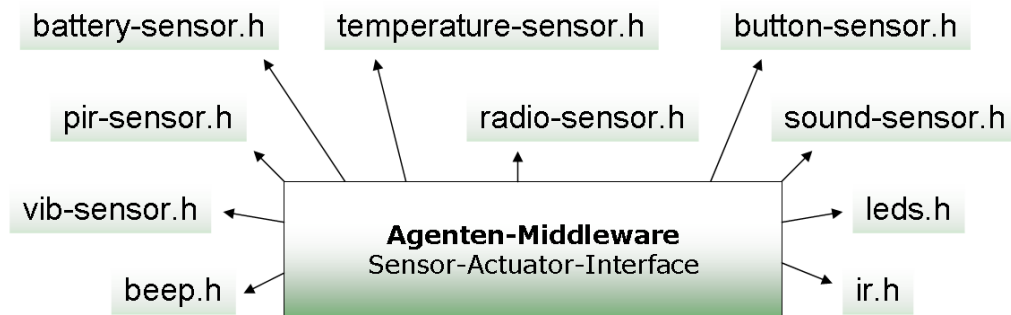


Figure 6.5: Access of sensor and actuators of the ESB430 in UbiMASS

of the modular concept of UbiMASS. It is only necessary to link the firmware components to the sensor-actuator interface of the new hardware.



### 6.1.4 Communication Layer

The agent middleware in UbiMASS is positioned on top of the communication layer that is responsible for the transfer of messages between sensor nodes. The communication bases on the *User Datagram Protocol (UDP)* a minimal and connectionless net-protocol. During the migration process an agent is fragmented into small packets. Each data packet has a size of 96 bytes and gets an additional header of 8 bytes. The UDP header contains four 16 bit fields as shown in Figure 6.6:

- *ID* field identifies the current agent. It is set randomly at the beginning and is incremented for each new agent.
- The *type* can get two possible values: a usual data packet has the type *TYPE\_DATA*, whereas a response packet that indicates a packet loss has the type *TYPE\_NACK*.
- Address field is used to reassemble the agent. The address of first packet gets the value of memory address the agent on the source was stored on. Each succeeding packet increases the value by the data length (96 Bytes). Destination host can use this information to reassemble packets in the right order.
- *Length* field shows the overall length of the sent data. This is used to check completeness of the entire data.

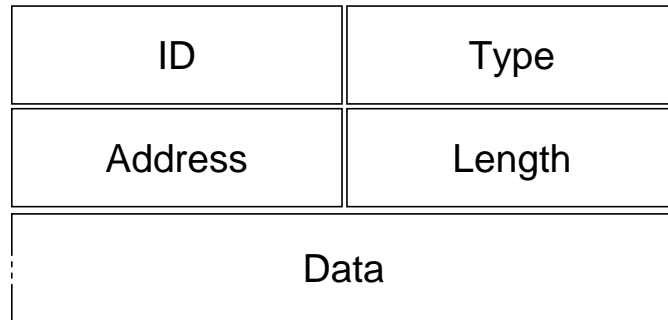


Figure 6.6: UDP datagram format in UbiMASS

Since UDP is a connectionless and non-reliable transfer protocol, there are no guarantees that a sent packet reaches the receiver or that packets arrive in a sorted order. The communication layer of UbiMASS closes this gap by checking the address and ID fields in the packet header. If there are missing or incorrect packets, UbiMASS requests them by sending a *NACK* message that contains the address of incorrect packet. UbiMASS ensures in this way a reliable communica-

tion between sensor nodes.

### 6.1.5 UbiMASS Agent

The core of an agent system is the agent itself. Agents in UbiMASS can be easily implemented using the common C language. Figure 6.7 illustrates an example of a UbiMASS agent. This agent has the simple task of activating the

```
1  #include "ubimass-esb.h"
2  #include "migration/migration-tmp.h"
3  #include <stdio.h>
4  #include <stdlib.h>
5  static struct etimer timer;
6  int i = 0;
7  int count_beeps = 2;
8
9  demand_init_data();
10 etimer_set(&timer, CLOCK_SECOND*4);
11 if (init_ok() == 1) {
12     count_beeps = get_init_data(0);
13 }
14 count_beeps++;
15 set_init_data(0, count_beeps);
16
17 while(i < count_beeps) {
18     etimer_reset(&timer);
19     beep_long(CLOCK_SECOND/4);
20     i++;
21 }
22
23 int len = migration_getLen();
24 start_migration(len);
25 }
```

Figure 6.7: Example of a UbiMASS agent

beeper on the current sensor board. The count of beeps is increased with each migration. Therefore, the agent uses variable `count_beeps`. After arriving on the current node agent has to request variable values from the previous node. This is shown on line 9 of the code, where the agent calls the middleware function `demand_init_data()`. After a waiting time in which the agent host requests variable values from the source host, the agent can check the status by calling

`init_ok()`. If variables are received correctly, the agent can access values over the function `get_init_data(id)`. After increasing the number of beeps, agent stores a new value by calling `set_init_data(id, value)` in order to be able to access the variable on the next node. The migration begins after calling the function `start_migration()`.

## 6.2 Evaluation

We have evaluated UbiMASS in several scenarios using real sensor boards. In order to illustrate the advantage of UbiMASS compared to traditional over the air reprogramming, we chose a scenario where the sensor nodes were attached in the way that the entire network forms a ring. It is assumed that in the middle of the ring there is an object (e.g. a mountain) that blocks the wireless connectivity, so that each node has only a single predecessor and successor.

In order to demonstrate reprogramming of the sensor network, the agent that has the simple task to count the number of sensor nodes is started first. The agent hops from node to node and increases the number of visited hosts. Since the network forms a ring, the agent will finally arrive at the first node where it displays the number of nodes on a connected PC. The round trip traverse of this agent is shown in Figure 6.8.

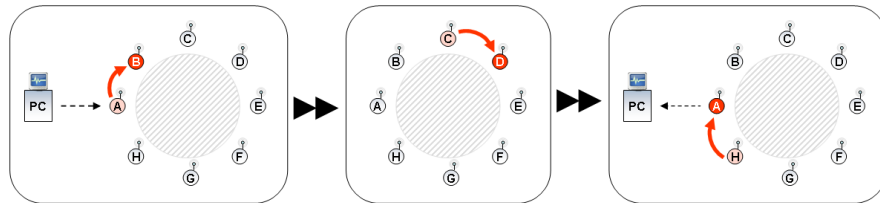


Figure 6.8: Round trip of a mobile agent in UbiMASS

Since we want to demonstrate the service switch, we start a second agent that accesses the light sensor on each node (e.g. to check the weather situation). A great advantage of UbiMASS in this context is that the second agent can be started with a short delay after the first agent. In this way, the agents provide different sensor information without losing much time. Figure 6.9 visualizes this specific scenario. UbiMASS offers agents the possibility to work asynchronously and autonomously. Since the agent carries all the information and sensor values, there is no need for additional communication except for the migration. After migration of the agent, the sensor node can perform different services by allowing other agents to run on it. It is interesting to compare this feature with other

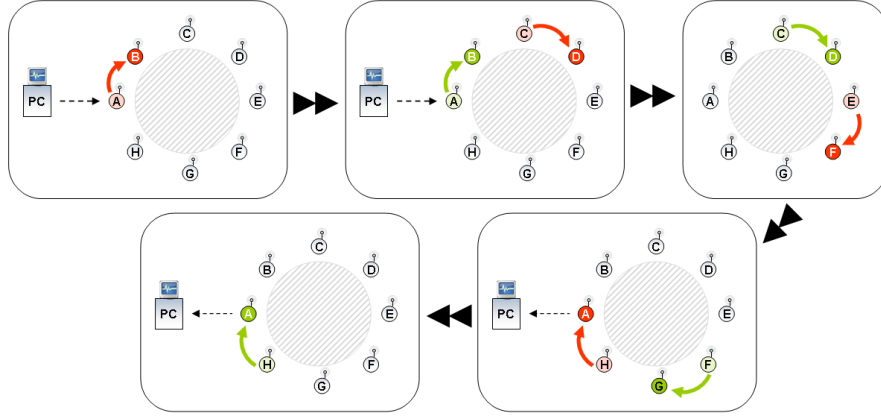


Figure 6.9: Round trip of two successive mobile agents in UbiMASS

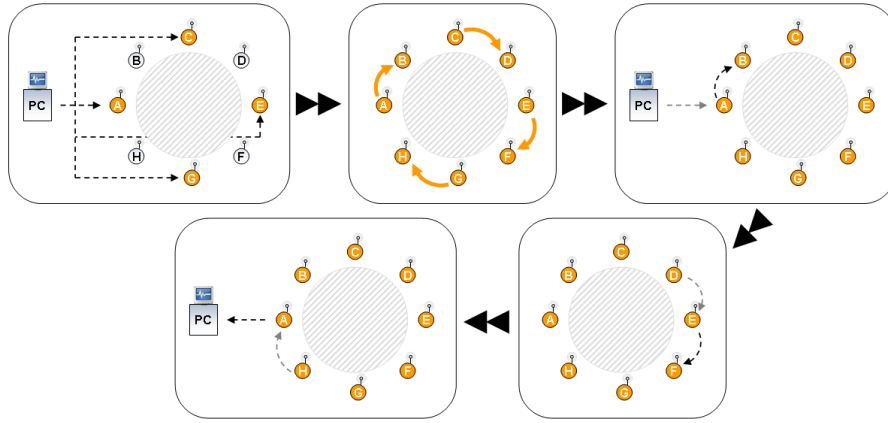


Figure 6.10: Round trip in Contiki

reprogramming systems. To investigate this, we implemented the same scenario with two successive agents as mentioned above on the open source operating system Contiki. With the specific application of *code propagation and storage* of Contiki it is possible to reprogram sensor nodes. After loading a program from a PC on a sensor node this program can be transferred to neighboring nodes by broadcasting the code. But this is limited to only one time, i.e. nodes that are reachable over two or more hops cannot be programmed with this approach. In our ring scenario, this leads to an enormous effort since we have to attach a PC at each second sensor node. After this work, it is necessary to send a round-trip message in order to receive the sensor values. Loading a second service requires the same effort again from the beginning. Figure 6.10 describes the required steps in Contiki compared to round trip of a single agent in UbiMASS (see Figure 6.8).

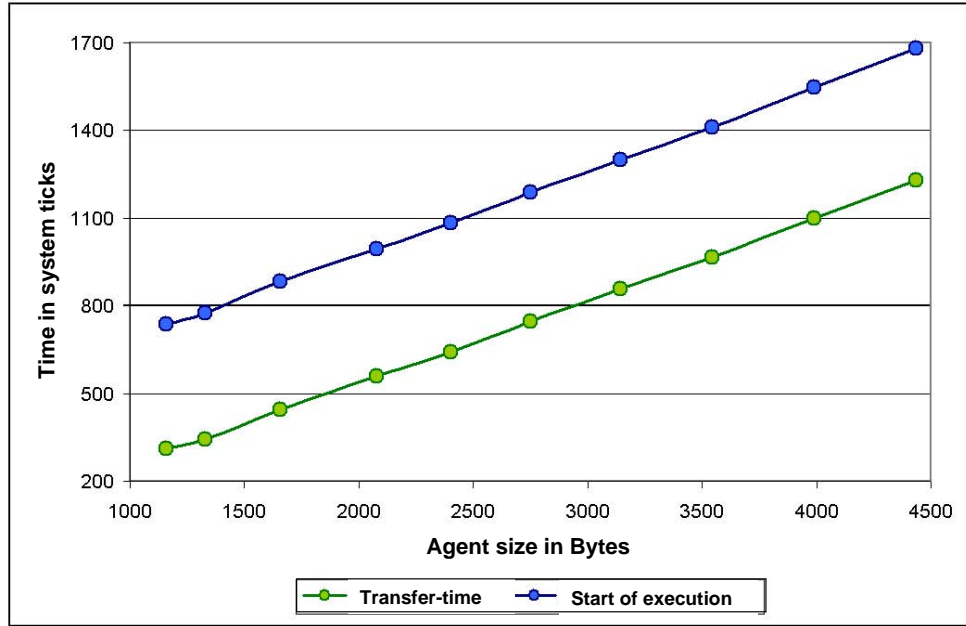


Figure 6.11: Receiving an agent: only transfer time vs loading time

In order to evaluate the transfer time and loading time of agents, we measured the communication between two neighboring nodes. Transfer time is only the time to send the packets from one node to another without considering the dynamic loading time of the agent. Loading time contains also the time for starting the agent on the destination node. Figure 6.11 shows a chart where the size of the agent is increased and the respective time is measured in system ticks (since the timer functions of the firmware provide system ticks). As expected the time increases linearly according to the size of the agent. Also the loading time is linear because the ELF loader has to handle more code if the agent size increases.

It is also interesting to compare the loading time of UbiMASS to on the air programming of Contiki. Figure 6.12 visualizes the results of this comparison. The chart clearly shows that the communication layer of UbiMASS works more efficiently. UbiMASS takes much less time to load the new service. Increasing the size of the code also increases the distance between both lines.

For the next scenario we implemented an agent that migrated to another node and back. We measured the round trip time in system ticks. It was also interesting to investigate the communication in relation to the distance between nodes. We performed several measurements by changing the location of the nodes. Figure 6.13 shows that increasing the distance leads to more packet losses which results in higher transfer times. Also obstacles like persons or walls and other interferences highly affect the communication time.

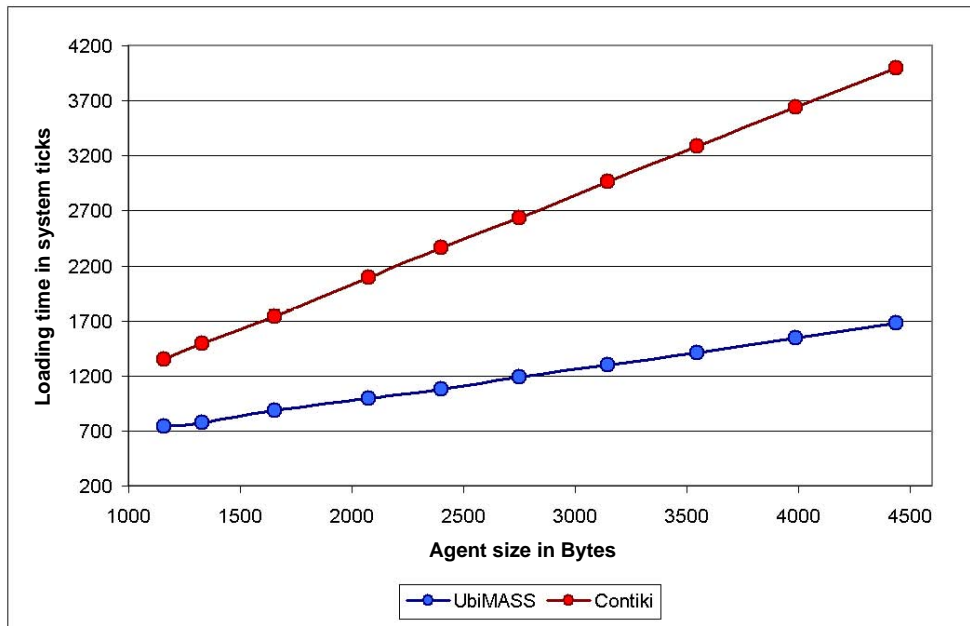


Figure 6.12: Receiving an agent: comparison with reprogramming in Contiki

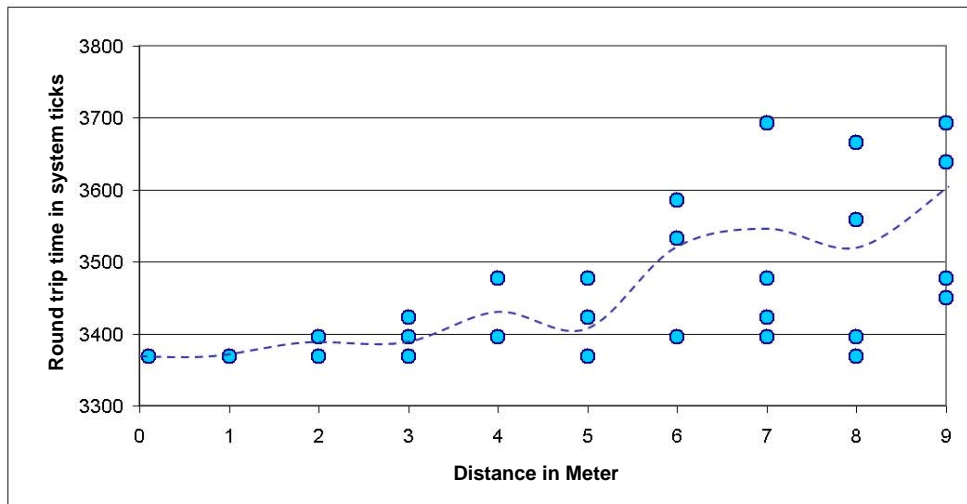


Figure 6.13: Round trip time subject to distance between sensor nodes



## 7 Security Architecture for WSN

Wireless sensor networks are highly vulnerable to attacks because they consist of numerous resource-constrained devices and communicate via wireless links. These vulnerabilities are exacerbated when WSNs have to operate unattended in a hostile environment such as battlefields. In such an environment, an adversary poses a physical threat to all the sensor nodes. An adversary may capture any node, compromising critical security data. Consequently, it is necessary to provide security services to these networks to ensure their survival. In general, there are two different terms typically used to express the vulnerabilities in security: threats and attacks. *Threat* is a potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm, i.e. a threat is a possible danger that might exploit vulnerability. *Attack* is an assault on system security that derives from an intelligent threat, i.e. an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system. Or in other words, a security threat is the expressed potential for the occurrence of an attack, while a security attack is an action taken against a target with the intention of doing harm.

### 7.1 Security Requirements in WSN

Sensor networks are used in a number of domains that handle sensitive information. Due to this, there are many considerations that should be investigated and are related with protecting sensitive information transferred between nodes (which are either sensor nodes or the base station) from being disclosed to unauthorized third parties. The comprehensive security requirements can be classified as follows:[RY06], [CKM00]

- **Confidentiality:** Confidentiality requirement is needed to ensure that sensitive information is well protected and not revealed to unauthorized third parties. The confidentiality objective is required in sensors environment to protect information traveling between the sensor nodes of the network or between the sensors and the base station from disclosure, since an adversary having the appropriate equipment may eavesdrop on the communication.



Compromised nodes are a big threat to confidentiality since the adversary could steal critical data stored on nodes such as cryptographic keys used to encrypt messages.

- **Authentication:** As in conventional systems, authentication techniques verify the identity of the participants in a communication, distinguishing in this way legitimate users from intruders. In the case of WSN, it is essential for each sensor node and base station to have the ability to verify that the data received was really sent by a trusted sender and not by an adversary that tricked legitimate nodes into accepting false data. If such a case happens and false data are supplied into the network, then the behavior of the network could not be predicted and probably will not outcome as expected. Authentication objective is essential to be achieved when clustering of nodes is performed. Clustering involves grouping nodes based on some attribute such as their location, sensing data, etc. Each cluster usually has a cluster head (CH) that is the node that joins its cluster with the rest of the WSN (meaning that the communication among different clusters is performed through CHs). In these cases, where clustering is required, two authentication situations should be regarded; first it is critical to ensure that the nodes belonging to each cluster will exchange data only with the authorized nodes and which are trusted based on some authentication protocol. Otherwise, if nodes within a cluster receive and process data from nodes that are not trusted within the current community, the expected data would be falsified and may cause damage. The second authentication situation involves the communication between CHs of each cluster. Communication must be established only with CHs that can prove their identity. No malicious node should be able to masquerade as a CH and communicate with a legitimate CH, sending it false data or either compromising exchanged data.
- **Integrity:** Integrity objective means the danger that information could be altered when exchanged over insecure networks. Lack of integrity could result in many problems since the consequences of using inaccurate information could be disastrous, for example for the health care sector where lives are endangered. Integrity controls must be implemented to ensure that information will not be altered in any unexpected way. Many sensor applications such as pollution and health care monitoring rely on the integrity of the information to function with accurate outcomes; it is unacceptable to measure the magnitude of the pollution caused by chemicals waste and find out later on that the information provided was improperly altered by the factory that was located near by the monitored lake. Therefore, there is urgent need to make sure that information is traveling from end to end without being intercepted and modified in the process.

- **Data Freshness:** One of the many attacks launched against WSN is the message replay attack where an adversary may capture messages exchanged between nodes and replay them later to cause confusion to the network. Data freshness objective ensures that messages are fresh, meaning that they obey in a message ordering and have not been reused. To achieve freshness, network protocols must be designed in a way to identify duplicate packets and discard them preventing potential mix-up.
- **Secure Management:** Management is required in every system that is constituted from multi components and handles sensitive information. In the case of WSN, secure management on base station level is needed. Since sensor node communication ends up at the base station, issues like key distribution to sensor nodes in order to establish encryption and routing information need secure management. Furthermore, clustering requires secure management as well, since each group of nodes may include a large number of nodes that need to be authenticated with each other and exchange data in a secure manner. In addition, clustering in each sensor network can change dynamically and rapidly. Therefore, secure protocols for group management are required for adding and removing members, and authenticating data from groups of nodes.
- **Availability:** Availability ensures that services and information can be accessed at the time that they are required. In WSN, there are many risks that could result in loss of availability such as sensor node capturing and denial of service attacks. Lack of availability may affect the operation of many critical real time applications like those in the health care sector that require a 24 / 7 operation that could even result in the loss of life. Therefore, it is critical to ensure resilience to attacks targeting the availability of the system and find ways to fill in the gap created by the capturing or disablement of a specific node by assigning its duties to some other nodes in the network.
- **Quality of Service:** Quality of Service (QoS) objective is a big headache to security. With all the limitations in WSN, QoS becomes even more constrained. Security mechanisms must be lightweight so that the overhead caused for example by encryption must be minimized and should not affect the performance of the network. Performance and quality in WSN involve the timely delivery of data to prevent for example propagation of pollution and the accuracy with which the data reported match what is actually occurring in their environment.
- **Robustness:** When some sensor nodes are compromised, the entire network should not also become compromised. The quantitative value with which this requirement should be satisfied depends on the application.

## 7.2 Security Threats in WSN

The possible security threats in WSNs are identified as follows [UAJP02]:

- **Passive Information Gathering:** An adversary with powerful resources collecting information from WSNs if information is not encrypted.
- **Node Subversion:** Capture of a node may reveal its information including disclosure of cryptographic keys, hence compromising the whole WSN.
- **False Node:** Addition of a malicious node by an adversary to inject malicious data. False node would be computationally robust to lure other nodes to send data to it.
- **Node Malfunction:** A malfunctioning node will generate inaccurate data that would jeopardize the integrity of a WSN, especially when that node is a data-aggregating node, for example, a cluster-head.
- **Node Outage:** What happens when a cluster-head stops functioning? WSN protocols should be robust enough to mitigate the effects of node outages by providing an alternate route.
- **Message Corruption:** When contents of a message are modified by an attacker, it compromises the message integrity.
- **Traffic Analysis:** Even if the message transfer is encrypted in WSNs, it still leaves the high probability of analysis of communication patterns and sensor activities revealing enough information to enable an adversary to cause more malicious harm to the WSN.

## 7.3 Security Attacks in WSN

In general, network security attacks are typically classified into passive and active attacks. In passive attacks, an unauthorized party gains access to an asset without modifying its content or actively attacking or disrupting a network. Passive attacks are very difficult to detect because they do not involve any alteration of the data, thus it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection. In active attacks, an unauthorized party makes modifications to a message, data stream, or file. It is possible to detect this type of attack, but it may not be preventable. There are two aspects of security attacks in WSN: layering based security attacks, and routing mechanisms attacks as shown in the next two subsections.

### 7.3.1 Layering-Based Security Attacks in WSN

#### 7.3.1.1 Application Layer Attacks

Data is collected and managed at application layer. Therefore, it is important to ensure the reliability of data. A resilient aggregation scheme that is applicable to a cluster-based network, where a cluster leader acts as an aggregator in WSN has been presented in [Wag04]. However, this technique is applicable, if the aggregating node is in the range with all the source nodes and there is no intervening node between the aggregator and source nodes. In the hierarchical clustering approach, a communication channel between the aggregator and base station has potentially limited bandwidth because the cluster leader as an aggregator itself is a sensor node [Wag04], [LH05]. To prove the validity of the aggregation, cluster leaders use cryptographic techniques to ensure data reliability.

#### 7.3.1.2 Transport Layer Attacks

The transport layer is responsible for managing end-to-end connections. Two possible attacks are possible in this layer: flooding and desynchronization.

- **Flooding:** Whenever a protocol is required to maintain state at either end of a connection, it becomes vulnerable to memory exhaustion through flooding [WS02]. An attacker may repeatedly make new connection requests until the resources required by each connection are exhausted or reach a maximum limit. In either case, further legitimate requests will be ignored. One proposed solution to this problem is to require that each connecting client demonstrate its commitment to the connection by solving a puzzle [WS02]. The idea is that a connecting client will not needlessly waste its resources creating unnecessary connections. Given that an attacker does not likely have infinite resources, it will be impossible for him/her to create new connections fast enough to cause resource starvation on the serving node. While these puzzles do include processing overhead, this technique is more desirable than excessive communication.
- **Desynchronization:** Desynchronization refers to the disruption of an existing connection [WS02]. An attacker may for example repeatedly spoof messages to an end host, causing that host to request the retransmission of missed frames. If timed correctly, an attacker may degrade or even prevent the ability of the end hosts to successfully exchange data, thus causing them to waste energy by attempting to recover from errors which never really existed. A possible solution to this type of attack is to require authentication of all packets communicated between hosts. Provided that the authentication method is itself secure, an attacker will be unable to send

the spoofed messages to the end hosts.

### 7.3.1.3 Network Layer Attacks

The network layer is responsible for routing of messages from node to node, node to CH, CH to other CHs, CHs to the base station, and vice versa. Routing protocols in WSN are based on two types of techniques: (1) ID-based protocols, in which packets are routed to the destination based on the IDs specified in the packets, and (2) data-centric protocols [GACE04] in which packets contain attributes that specify the type of data being provided. The attacks in the network and the routing layer are described in the following [KW03]:

- **Routing Loops or Spoofed/Fake Routing Information:** In WSN, routing loops attack the information exchanged between nodes. False error messages are generated when an attacker alters and replays the routing information. Routing loops attract or repel the network traffic and increase node-to-node latency. A countermeasure against spoofing and alteration is to append a Message Authentication Code (MAC) after the message. By adding a MAC to the message, the receivers can verify whether the messages have been spoofed or altered. To defend against replayed information, counters or timestamps can be included in the messages [HPJ03].
- **Selective Forwarding:** Selective forwarding is a way to influence the network traffic by believing that all the participating nodes in network are reliable to forward the message. In selective forwarding attack, malicious nodes simply drop certain messages instead of forwarding every message. Once a malicious node cherry picks on the messages, it reduces the latency and deceives the neighboring nodes that they are on a shorter route. Effectiveness of this attack depends on two factors. First is the location of the malicious node. The closer it is to the base station, the more traffic it will attract. Second is the percentage of messages it drops. When a selective forwarder drops more messages and forwards less, it retains its energy level, thus remaining powerful to trick the neighboring nodes. One defense against selective forwarding attacks is using multiple paths to send data. A second defense is to detect the malicious node or assume it has failed and seek an alternative route [HPJ03].
- **Sinkhole Attack:** In sinkhole attacks, an adversary attracts the traffic to a compromised node. The simplest way of creating a sinkhole is to place a malicious node where it can attract most of the traffic, possibly closer to the base station or malicious node itself, which deceptively acts as a base station. One reason for sinkhole attacks is to make selective forwarding possible to attract the traffic toward a compromised node. The nature of

WSNs where all the traffic flows toward one base station makes this type of attack more susceptible.

- **Sybil Attack:** This is a type of attack where a node creates multiple illegitimate identities in WSNs either by fabricating or stealing the identities of legitimate nodes [JNSP04]. Sybil attacks can be used against routing algorithms and topology maintenance; it reduces the effectiveness of fault-tolerant schemes such as distributed storage and disparity. Another malicious factor is geographic routing where a Sybil node can appear at more than one place simultaneously.
- **Wormholes:** In wormhole attacks, an adversary positioned closer to the base station can completely disrupt the traffic by tunneling messages over a low-latency link [MNP04]. Here an adversary convinces the nodes which are multihop away that they are closer to the base station. This creates a sinkhole because an adversary on the other side of the sinkhole provides a better route to the base station.
- **Hello Flood Attack:** This involves broadcasting a message with stronger transmission power and pretending that the HELLO message is coming from the base station [HMH06]. Message receiving nodes assume that the HELLO message sending node is the closest one and they try to send all their messages through this node. In this type of attack, all nodes will be responding to HELLO floods and wasting the energies. The real base station will also be broadcasting the similar messages but will have only a few nodes responding to it.
- **Acknowledgment Spoofing:** Routing algorithms used in WSNs sometimes require Acknowledgments (ACK) to be used. An attacking node can spoof the ACK of overheard packets destined for neighboring nodes in order to provide false information to those neighboring nodes. An example of such false information is claiming that a node is alive when in fact it is dead.

#### 7.3.1.4 Data Link Layer Attacks

The data link layer ensures reliable point-to-point and point-to-multipoint connections in a communication network. Attacks at the link layer include purposely introduced collisions, resource exhaustion, and unfairness.

- **Collisions:** A collision occurs when two nodes attempt to transmit on the same frequency simultaneously. An adversary may strategically cause collisions in specific packets such as ACK control messages. TinySec [KSW04] has introduced link layer encryption, which depends on a key manage-

ment scheme. Protocols like Lightweight Medium Access Control (LMAC) [HH04] have better properties, which are viable countermeasures at this layer.

- **Exhaustion:** Repeated collisions can also be used by an attacker to cause resource exhaustion. Unless these hopeless retransmissions are discovered or prevented, the energy reserves of the transmitting node and those surrounding it will be quickly depleted. A possible solution is to apply rate limits to the MAC admission control such that the network can ignore excessive requests, thus preventing the energy drain caused by repeated transmissions. A second technique is to use time-division multiplexing where each node is allotted a time slot in which it can transmit.
- **Unfairness:** Unfairness can be considered a weak form of a Denial of Services (DoS) attack. Instead of preventing access to a service outright, an attacker can degrade it in order to gain an advantage such as causing other nodes in a real-time MAC protocol to miss their transmission deadline. The use of small frames lessens the effect of such attacks by reducing the amount of time an attacker can capture the communication channel.

#### 7.3.1.5 Physical Layer Attacks

The physical layer emphasizes the transmission media between sending and receiving nodes. The data rate, signal strength, and frequency types are also addressed in this layer. Following two vulnerabilities are explored:

- **Jamming:** Is a type of attack which interferes with the radio frequencies that a network's nodes are using [SP04]. A jamming source may either be powerful enough to disrupt the entire network or less powerful and only able to disrupt a smaller portion of the network. Even with lesser-powered jamming sources, such as a small compromised subset of the network's sensor nodes, an adversary has the potential to disrupt the entire network provided the jamming sources are randomly distributed in the network. Ideally, the Frequency Hopping Spread Spectrum (FHSS) is used in sensor networks to overcome the jamming attack. Code spreading is another technique used to defend against jamming attacks and is common in mobile networks.
- **Tampering:** Given physical access to a node, an attacker can extract sensitive information such as cryptographic keys or other data on the node. The node may also be altered or replaced to create a compromised node that the attacker controls. One defense to this attack involves tamper-proofing the node's physical package.

### 7.3.2 Routing Mechanism Attacks in WSN

Most routing mechanisms designed for WSN assume that nodes in the network do not misbehave. This provides many opportunities for malicious nodes to attack and disrupt the routing mechanism. These attacks can be classified also as passive and active attacks. A passive attack typically involves only eavesdropping the routing traffic, which can be solved either by using encryption or by transmitting parts of message over multiple disjoint paths and reassemble them at the destination as mentioned before. Active attacks can be classified into three classes according to the on time routing process as shown in the following subsections.

#### 7.3.2.1 Attacks on Route Discovery Process

Such attacks attempt to prevent other legitimate nodes from establishing routing paths by sending fake routing information. Moreover, a malicious node can send excessive route request messages to exhaust the network bandwidth. The former is to provide fake information to spoof the route discovery process, while the latter is to overly use the route discovery process. Both of them attempt to cause DoS.

#### 7.3.2.2 Attacks on Route Selection Process

Such attacks attempt to increase the chance that malicious nodes are selected other legitimate nodes as part of their routes. Four possible of attacks of this type are involved; Hello flood attacks, sinkhole attacks, wormhole attacks, and Sybil attacks.

#### 7.3.2.3 Attacks after Establishing Routing Paths

Once a source node establishes a route through a malicious node, the malicious node can unscrupulously drop the data packets from the source or modify the contents of packets if encryption is not applied. Two attacks are involved, blackhole attack and spam attack.

- **Blackhole Attack:** Here, the malicious nodes intentionally drop all received messages from the source to prevent these messages from being propagated to any further. To defend against such attack, routing protocols should have route maintenance mechanism such that if a node in a path finds that its next hop neighbor no longer propagates data packets, it would notify the source to recreate another path. Three mechanisms



are proposed to defend blackhole attack as follow [AKWK04]. The first approach is the Source Routing, in which the source specifies in each data packet the sequence of nodes that the packet has to traverse. Second is the Destination Acknowledgment, in which the destination sends back an ACK to the source along the same route (but in reverse direction) when it receives each data packet. Third mechanism is Timeouts, in which the source and each intermediate node set a timer for each data packet, during which they expect to receive an ACK from the destination or a Fault Announcement (FA) from other intermediate nodes. When the timer expires at a node, it generates an FA message and propagates this message back to the source. All data, ACK, and FA messages are authenticated by a MAC so that these messages cannot be modified or fabricated by a malicious node. Note that the source can detect the presence of a potential blackhole attacker, when it receives an FA message and thus select another route to forward its packets.

- **Spam Attack:** These type of attacks are caused by malicious nodes generating frequent dummy messages to specified targets in the network. A WSN is more vulnerable to this attack because the sink is usually the only target to be attacked. To defend against such attack, the Detect And Defend Spam (DADS) scheme proposes a concept of quarantine regions to isolate spam attackers [SCL04]. In DADS the remote sink is responsible for detecting whether there are spam attacks in the network by observing the packet generation rate of the overall sensor nodes in network.

## 7.4 SecSens - A Novel Security Architecture for Wireless Sensor Networks

Based on the investigated results of last sections, a novel security architecture for wireless sensor networks, called *SecSens*, will be introduced in the following [BUB10] [BUB09]. The sensor network in SecSens consists of clusters, each containing simple sensor nodes  $u_i$  and one powerful sensor node  $v$  that acts as a cluster-head. Sensor nodes  $u_i$  are connected directly to the cluster-head, because routing in clusters is not necessary. Sensor nodes can be a member of several clusters. Cluster-heads again build together an inter-cluster network, that is used to transfer messages to base stations. It is assumed that sensor nodes have a fixed position, once they are attached to a location. SecSens works with multiple base stations to avoid the risk of single-point-of-failure. The security architecture of SecSens combines several security approaches in order to provide high protection. Basically SecSens contains four components, which interact with each other: authenticated broadcasts, key management, routing, and en-route filtering.

### 7.4.1 Authenticated Broadcasts

Authenticated broadcasts ensure that the stated sender is identified as the true sender. Symmetric approaches use a shared key to generate a *message authentication code (MAC)*. In case of only one receiver, the sender is clearly identified. But if there are multiple receivers with the same shared key, this approach for authentication is not applicable. Potentially each receiver could be the sender. To solve this problem, there must exist an asymmetry between sender and receiver.

SecSens provides two authenticated broadcasts: broadcasts from base stations, and broadcasts in clusters. It is assumed that base stations are trustworthy and can not be infiltrated. In order to generate an asymmetry, SecSens uses key chains. Each packet contains a key. To decrypt a previous packet, a node has to wait for the key of next packet. (Figure 7.1).

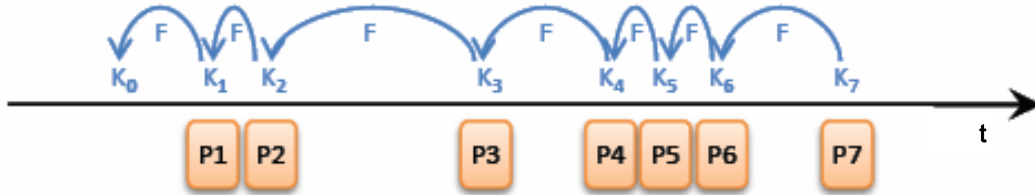


Figure 7.1: Key chain approach

The base station generates a key chain  $K_0^b \dots K_n^b$  with sufficient number of keys using a publicly known one-way function  $F$ , so that for  $i \in \{0, \dots, n-1\}$  is:

$$K_i^b = F(K_{i+1}^b) \quad (7.1)$$

Each node knows that the first key in the chain is  $\langle i, K_i^b \rangle$  with  $i = 0$ . Therefore,  $K_0^b$  is public and  $K_1^b$  is the first non-disclosed key. Keys can be used only once. In order to broadcast a message  $M$ , the base station calculates the corresponding MAC using the next non-disclosed key  $K_i^b$  and sends it together with used key index to all its neighbors:

$$BS \rightarrow * : \quad i, MAC_{K_i^b}(M) \quad (7.2)$$

A receiving node  $u$  checks, if it has already received a MAC for the stated index, before storing the MAC and index  $i$  into the MAC buffer. Consequently node  $u$  accepts only one MAC per key index. If  $i$  is a new index and  $u$  is a cluster-head, it forwards the message to all its neighbors. In this way, it is efficiently

distributed over the inter-cluster network to all sensor nodes. After maximum time  $T$  all sensor nodes know the MAC together with its key index. Sensor nodes can not manipulate the MAC, because the base station has not yet disclosed  $K_i^b$  at this time. Time  $T_p$  describes a dynamically adaptable system parameter. The base station can set  $T_p$  depending on the network size, whereas  $T < T_p$ . After expiration of  $T_p$ , the base station sends the actual message  $M$  besides the disclosed key data  $\langle i, K_i^b \rangle$  to all neighbors:

$$BS \rightarrow * : \quad M, \langle i, K_i^b \rangle \quad (7.3)$$

Sensor node  $u$  can now verify  $K_i^b$  using a previous disclosed key  $K_{i-1}^b$ . If it does not have  $K_{i-1}^b$ , it can verify the current key through recursively performing Equation 7.1 with previous keys. To prevent DoS-attacks, the new key must not be older than  $G_{max}$  generations. If  $K_i^b$  is finally verified, node  $u$  makes new key data  $\langle i, K_i^b \rangle$  effective. Subsequently,  $u$  can check message  $M$  using  $MAC_{K_i^b}(M)$  and index  $i$ . It is important that the index for MAC and key are the same.

The concept of authenticated broadcasts for base stations has a disadvantage, if it is used for local authenticated broadcasts between clusters, because of the delayed disclosure of keys. However, all receivers of a message in a cluster can be reached after a single hop. This fact can be used to simplify the concept, in order to avoid time delays. The cluster-head generates a key chain  $K_0^b \dots K_n^b$  using a publicly known function  $F$ . All cluster members receive the first key  $K_0^b$  over a secure connection using pair-wise shared keys known only by cluster-head and respective node. Cluster-head  $v$  uses for each local authenticated broadcast a key  $K_i^b$ , which is not yet disclosed. It sends the message  $M$  together with the key data:

$$v \rightarrow * : \quad M, K_i^b \quad (7.4)$$

Cluster nodes can verify  $K_i^b$  and the message  $M$  using previously disclosed keys. With this the sender is authenticated as cluster-head, because only the cluster-head can know  $K_i^b$ , that was not public until current message. On the other hand, no intermediate node can manipulate the message  $M$ , because all potential receivers of  $M$  are reachable with only one hop.

### 7.4.2 Key Management

Sensor nodes basically distrust each other. To build trust between two or more nodes, a shared secret in the form of keys is needed. However, neighborhood and relations between sensor nodes are not known before. Therefore, nodes must build trust during life-time, more strictly in the initial phase. For security reasons,

a sensor node should join a network only once. In this critical phase, it can establish shared keys with its neighbors. Since this procedure is performed only once, the node is binding itself to the location. Furthermore, sensor nodes can communicate at several levels, that is cluster- or network-wide. Consequently, SecSens uses several kinds of keys to fulfill different security requirements.

*Activation:* In the initial phase, a sensor node needs an initial key  $K_I$ . This key is stored only on a specific activation node, which does not take part for usual networking tasks. The basic idea is that sensor nodes can not install themselves independently. Instead a trustworthy employee, who owns the activation node, establishes sensors. In order to add a set of sensor nodes, the base station stores a randomly generated master key  $K_A$ , timer  $T_A$ , initial key  $K_I$ , and current group key  $K^g$  onto the activation node  $A$ . Using master key  $K_A$ , the base station can generate for each sensor node  $u$  a personal activation key  $K_u^a$  based on the node ID.

$$K_u^a = F_{K_A}(u) \quad (7.5)$$

All key material and other security critical data are stored only in RAM of  $A$ .  $T_A$  determines period of validity for the master key. After expiration of this time, the activation node deletes  $K_A$  and all critical data. Each sensor node  $u$  has a unique ID and a personal activation key  $K_u^a$ . In order to activate sensor node  $u$ , the activation node  $A$  has to be in the communication range. For security reasons, the radio power of  $A$  is kept low, to ensure physical proximity. After turning on for the first time, sensor node  $u$  broadcasts periodically its plain ID and the ID encrypted with the personal key with the same low radio power. Activation node  $A$  can easily verify the ID, because it knows the master key  $K_A$ . As the next step,  $A$  encrypts with personal key of  $u$  the initial key  $K_I$ , group key  $K^g$ , and data  $X$  that was given by base station. Finally,  $A$  sends encrypted message to sensor node  $u$ :

$$u \rightarrow A : \quad u, MAC_{K_u^a}(u) \quad (7.6)$$

$$A \rightarrow u : \quad \{K_I, K^g, X\}_{K_u^a} \quad (7.7)$$

After receiving all key material, sensor node  $u$  is activated and it deletes the personal activation key.

*Group keys:* The group key  $K^g$  is used by the base station to secure network-wide communication. An attacker, who compromises a node, can also access the group key. In order to update  $K^g$ , the base station broadcasts first a list of known compromised nodes  $\{x_1, \dots, x_m\}$  to all sensor nodes. Additionally, it sends a verification key  $F_{K^{g'}}(0)$ , whereas  $K^{g'}$  is a new randomly generated group key, and  $F$  a publicly known one-way function.  $F_{K^{g'}}(0)$  is used later to verify the new group key  $K^{g'}$ .

$$BS \rightarrow * : \quad i, MAC_{K_i^b}(\{x_1, \dots, x_m\} || F_{K^{g'}}(0)) \quad (7.8)$$

$$BS \rightarrow * : \quad \{x_1, \dots, x_m\}, F_{K^{g'}}(0), \langle i, K_i^b \rangle \quad (7.9)$$

The base station uses an authenticated broadcast with key  $K_i^b$  and index  $i$  that is not disclosed yet. After receiving key  $K_i^b$  and successfully verifying the above message (see section 7.4.1), sensor nodes delete all pair-wise shared keys or cluster-keys with compromised node  $x_i$ . Cluster-heads additionally update their cluster-keys and inform other non-compromised cluster-heads about new cluster-key. Afterwards, all sensor nodes store verification key  $F_{K^{g'}}(0)$ . As a second step, the base station publishes new group key  $K^{g'}$ . Therefore, it encrypts the group key using its cluster-key  $K_{BS}^c$  and transfers the message to all direct neighbors. The neighbors can verify  $K^{g'}$  using verification key  $F_{K^{g'}}(0)$  and store afterwards the new group key.

$$BS \rightarrow * : \quad \{K^{g'}\}_{K_{BS}^c} \quad (7.10)$$

If receiver  $u$  is a cluster-head, it forwards a new group key  $K^{g'}$  encrypted by its own cluster key  $K_u^c$ . Consequently, the new group key  $K^{g'}$  is forwarded over the inter-cluster network to all sensor nodes. Since cluster-heads have updated their cluster keys before, the compromised nodes do not receive the new group key. This procedure is periodically repeated by base stations to prevent the network against attacks. If there are no new known compromised nodes, the transferred list is empty.

*Pair-wise shared keys:* For secure communication between sensor nodes, pair-wise shared keys are used. A new cluster-head exchanges a pair-wise key with all neighbors. Simple sensor nodes communicate only over cluster-head, therefore, they need only a shared key with their cluster-head. Using the initial key  $K_I$  each node  $u$  generates a personal master key  $K_u^p$  based on its ID. In order to establish a pair-wise shared key with its neighbor  $v$ , node  $u$  needs the ID of  $v$ . For this reason  $u$  broadcasts a HELLO-message containing its ID. If  $v$  decides to establish secure connection with new node  $u$ , it answers with an acknowledgment containing its own ID.

$$u \rightarrow * : \quad u \quad (7.11)$$

$$v \rightarrow u : \quad v, MAC_{K_v^p}(u||v) \quad (7.12)$$

The additional MAC authenticates the acknowledgement of  $v$ , because  $u$  can calculate master key  $K_v^p$  of  $v$  using initial key  $K_I$ . Node  $u$  does not need to authenticate itself, because the succeeding message exchange verifies the identity of  $u$ . The pair-wise shared key  $K_{uv}^p$  can be calculated by both nodes without new message exchange:

$$K_{uv}^p = F_{K_v^p}(u) \quad (7.13)$$

After expiration of time  $T_I$  the nodes delete initial key  $K_I$  and all personal master keys of their neighbors received during initialization. Only one personal master key is stored for future pair-wise keys with new sensor nodes. The annulment of compromised pair-wise keys is efficiently realized by deletion of corresponding keys.

*Cluster keys:* Sensor nodes transfer information to all other cluster members using the cluster key without encrypting the message for each receiver separately. This approach allows in-network-processing and passive participation of sensor nodes within a cluster. Cluster-head  $u$  generates randomly cluster key  $K_u^c$ , if  $u$  joins a network or if  $u$  updates cluster key because of compromised nodes. Each cluster member  $v_1, \dots, v_m$  receives new cluster key  $K_u^c$ , whereas  $u$  encrypts cluster key using pair-wise shared keys  $K_{uv_i}^p$  for  $i \in \{1, \dots, m\}$ :

$$u \rightarrow v_i : \quad \{K_u^c\}_{K_{uv_i}^p} \quad (7.14)$$

Only sensor node  $v_i$  can decrypt cluster key and store it. If an additional sensor node  $v$  joins the network, it establishes a new pair-wise shared key with cluster-head. In this case it also gets the current cluster key. If a cluster member is compromised, cluster-head annuls cluster key  $K_u^c$  and distributes new key  $K_u^{c'}$  as described above, without sending it to the compromised nodes.

### 7.4.3 Routing

To prevent attacks on routing level or restrict them locally, SecSens provides a secure routing protocol. Simple sensors in SecSens do not need a routing capability, because they exclusively communicate with the cluster-head. Therefore, routing is used only within the inter-cluster network built by cluster-heads. The routing algorithm has two phases: initialization and the actual routing. In the initialization phase each node gets a level using breadth first search that determines the distance to base station in hops. Since base station has level 0, its direct neighbors have level 1. The base station uses an authenticated broadcast including its ID  $BS_u$ , a non-disclosed key  $K_i^b$ , and key index  $i$  to authenticate an initialization.

$$BS \rightarrow * : \quad i, MAC_{K_i^b}(BS_u) \quad (7.15)$$

$$BS \rightarrow * : \quad BS_u, \langle i, K_i^b \rangle \quad (7.16)$$

Cluster-heads can identify from authenticated messages, which base station wants to update routing information. After reception of initialization, cluster-heads have time  $T$  to modify their routing tables. After expiration of  $T$ , further changes are not allowed. Cluster-heads set their level on  $L = \infty$  after reception of the

message (see Equation 7.16). The breadth first search can now begin. Starting from the base station, the level values are locally broadcasted by cluster-heads. To prevent outsider-attacks each cluster-head  $u$  uses key  $K_i^b$ , that will be published later, and its cluster key  $K_u^c$  to generate an encrypted message containing ID and level value  $L_u$ :

$$u \rightarrow * : \quad \{u || L_u\}_{K_u^c}, K_i^b \quad (7.17)$$

A cluster-head  $v$  updates its level to  $L_v = L_u + 1$ , if  $L_v > L_u + 1$ . It also stores level of  $u$ . After level update,  $v$  forwards its level value to its neighbors in the same way. Each base station triggers its own initialization without disturbing ongoing updates of others. The cluster-heads manage a routing table for each base station.

SecSens uses probabilistic multi-path routing based on the level values to forward messages from cluster-heads on the way to the corresponding base station. Cluster-heads build up a trust matrix, where each transmission to its neighbors is recorded. Based on this trust information and current level, cluster-heads calculate a probability value and write it into the packet header. This value is used to decide in which direction the packet has to be send. Each cluster-head modifies the probability value and sends the message over the most trustworthy route. Since this could lead to the problem that a packet stays at the same level while making a round-trip, the weight of upper level increases with each hop. This ensures that packet transfer goes in the direction of the base station.

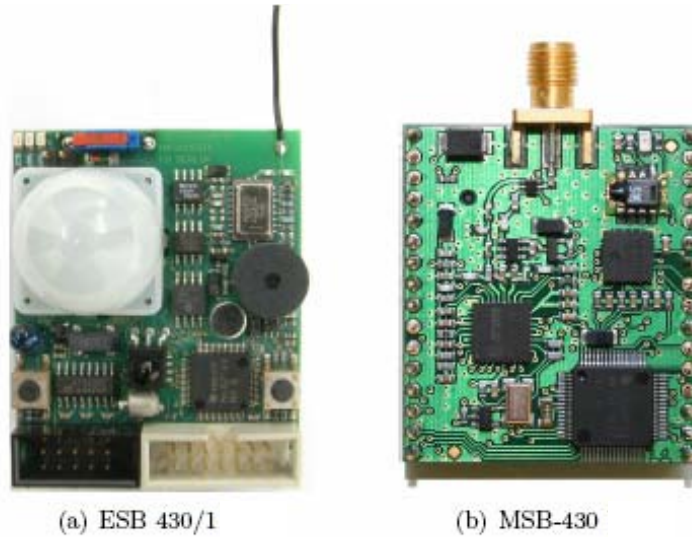


Figure 7.2: Sensor boards: ESB 430/1 and MSB-430

	AB	KM	MPR	EF	total
cluster-head: RAM	434 B	9 B	710 B	41 B	1194 B
cluster-head: EEPROM	800 B	2440 B	252 B	981 B	4473 B
sensor node: RAM	54 B	9 B	0 B	0 B	63 B
sensor node: EEPROM	0 B	92 B	0 B	537 B	629 B

Table 7.1: SecSens memory requirements, (AB) - authenticated broadcasts, (KM) - key management, (MPR) - multi-path routing, (EF) - en-route filtering

Furthermore, SecSens provides passive participation, i.e. sensor nodes listen to packet transmissions of their neighbors. If cluster-head  $u$  detects a packet addressed to its neighbor  $v$ , and recognizes that  $v$  is not forwarding the message,  $u$  takes responsibility with a certain (low) probability. Also, if  $u$  assumes that  $v$  forwards the message to a non-existent node,  $u$  takes care of transferring.

#### 7.4.4 En-route Filtering

Attacks like *report fabrication* or *false data injection* threaten the network by manipulating and infiltrating sensor data. SecSens prevents such threats using en-route filtering extending approach of [YLLZ05]. Cluster-heads generate data reports containing sensor information of cluster members for sending them to base stations. These reports are verified during transfer through the inter-cluster network. En-route filtering consists of three phases: key generation, report generation, and verification.

*Key generation:* SecSens provides a global pool containing  $N$  en-route keys  $\{K_0^e, \dots, K_{N-1}^e\}$ . The keys  $K_i^e$  are subdivided in  $n$  partitions with each  $m$  keys. Each sensor node generates all en-route keys in the initial phase using one-way function  $F$  and chooses randomly a partition  $j$  where it finally draws  $k < m$  keys from set  $j$ :

$$\forall i \in \{0, \dots, N-1\} : K_i^e = F_{K_M^e}(i) \quad (7.18)$$

Only base stations can definitely detect all fault reports, because they possess all keys of the key pool. Based on the same partition  $j$ , each node calculates in a similar way location key  $K_{C,j}^l$  for all its clusters that it senses as a member. Sensor nodes bind themselves locally to actual cluster by the location key. After the initial phase nodes delete all remaining unused keys.

*Report generation:* If a cluster-head wants to generate a report, it collects sensor data from all its cluster members. Sensor nodes belonging to the same cluster report events (sensor data) collectively by generating MACs based on their en-



route keys, whereas keys must be chosen from different partitions. These multiple MACs collectively act as the proof that a report is legitimate. Finally, the cluster-head forwards the report to the base station over the inter-cluster network.

*Verification of reports:* A cluster-head receiving a report checks, if it has one of the keys, that were used to generate the MACs in the report. With a certain probability, it will be able to verify the correctness of MACs. A report with an insufficient number of MACs will not be forwarded. A compromised node has keys from one partition and can generate MACs of one category only. Since keys and indices of distinct partitions must be present in a legitimate report, the compromised node has to forge the other key indices and corresponding MACs. This can be easily detected by cluster-heads possessing these keys. If a cluster-head has none of the keys and number of MACs is correct, it forwards report to the next cluster-head. Even if a forged report reaches a base station, it can be detected, because base stations know all the used keys.

#### 7.4.5 Evaluation

SecSens was evaluated on an environment with different kinds of sensor nodes: ESB 430/1 and MSB-430 of Freie University Berlin. Both sensor boards have the TI MSP430 microcontroller. ESB 430/1 contains 60 KB flash memory and 2 KB RAM, whereas MSB-430 has 55 KB flash memory and 5 KB RAM. Because of the larger RAM, MSB-430 was used as cluster-head. We used four cluster-heads managing each five sensor nodes that were all ESB boards, making 20 nodes altogether. Two PCs act as base stations. We could successfully perform all stages of SecSens which were described before.

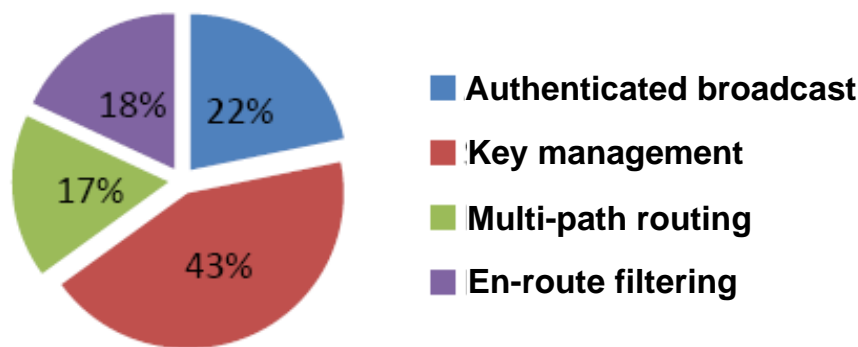


Figure 7.3: Memory map of SecSens security components

In order to use memory effectively, SecSens stores frequently changing data in

	packets	data	energy
sensor node	15	0,2 KB	0,2 Ws
cluster-head	142	4,1 KB	2,6 Ws

Table 7.2: Costs in initial phase

RAM and relatively static data in EEPROM. Table 7.1 describes memory consumption of cluster-heads and sensor nodes. The most memory space is used for key management which takes 43% of RAM. Figure 7.3 shows the memory map of SecSens security components.

In the initial phase, energy consumption is comparatively high. For establishing the network and distribution of keys, cluster-heads consume in average  $2.6Ws$  energy, whereas sensor nodes need only  $0.2Ws$  (see Table 7.2).

Energy consumption for sending reports depends on the distance between cluster-head and the base station. We established in a second test a network with up to 20 cluster-heads placed in a line side by side. The last cluster-head received level 20, which means that it needs 20 hops to reach the base station. We measured the energy consumption for sending reports from different levels. The multi-path routing ensures a robust transmission, but sending duplicated packets from several routes (fanout) increases the energy consumption. Figure 7.4 shows consumed energy for reports using no fanout (reference) or multiple fanouts.

As mentioned above, nodes can listen to packet transmission of neighbors and can take the responsibility for forwarding with a certain probability in case of detected failures. Figure 7.5 describes the energy results for passive participation with different probabilities.

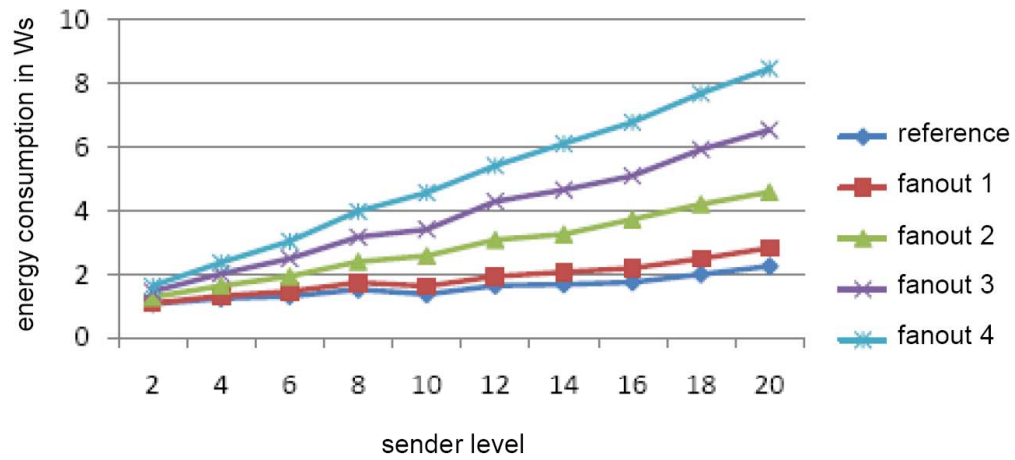


Figure 7.4: Consumed energy for reporting

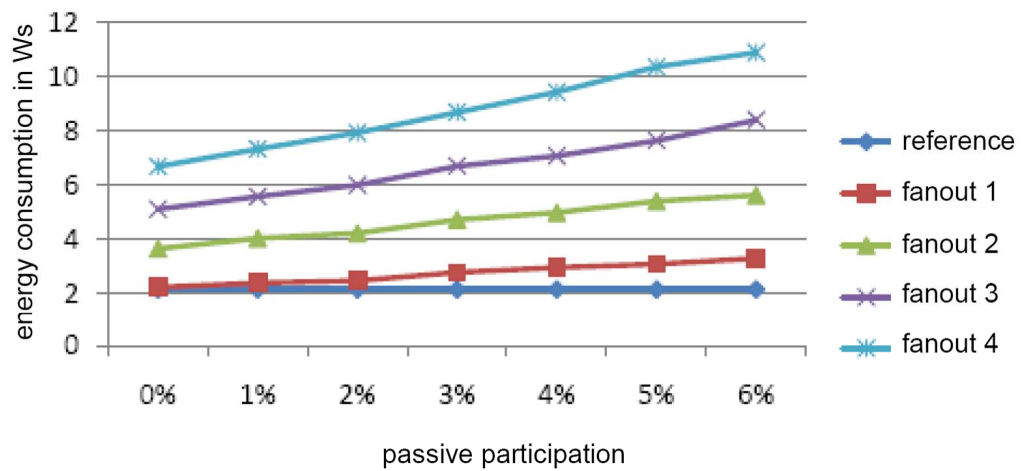


Figure 7.5: Energy consumption with passive participation

## 8 Location Estimation and Tracking in WSN

Recent developments in the area of ubiquitous and pervasive computing emphasize the interest in context-aware applications. Location is probably the most important context. Indoor location estimation and tracking remains challenging due to the lack of usable and cost-effective technologies. GPS has proved itself for outdoor usage, but it is not suitable for indoor applications due to poor coverage.

To gather real world information from the environment, ubiquitous and pervasive systems build on sensors attached in everyday objects. Wireless connectivity of sensors opens a wide range of applications. The main focus of this chapter is to use wireless sensor network for location estimation and tracking. Since the cost for sensors is low, the location tracking system is assumed to be cost-effective. Using wireless connections, it is possible to calculate current location of a user or an object. For some applications it is even sufficient to estimate user's location in a room. Providing more accuracy opens up an opportunity for more specific services. The Smart Doorplate project [BKS<sup>+</sup>08] [TBPU03] establishes a ubiquitous environment with intelligent doorplates, which present information about the office and employees to the visitors outside. Electronic touch-screen displays provide location of the office owner and some status information, such as, person is on the phone, absent, or busy. This chapter describes *LocSens* ([BUB08b], [BKSU09]) a location tracking system based on wireless sensors, that is used to get location information for Smart Doorplate services. LocSens works with fixed room sensors that communicate with mobile sensors carried by users.

Usually each sensor node in a wireless sensor network has its own processor, memory and application specific sensors. LocSens uses the ESB430 sensor boards that are developed at the Freie University of Berlin [Sca07].

It is a characteristic of sensor nodes that all resources are extremely limited. The energy supply is usually provided by a battery. LocSens uses cable operated sensor nodes for fixed positions in rooms as well as battery operated sensor boards carried by users. For current prototypes only a limited memory capacity is available. On the average sensor boards have less than 20 kilobytes of RAM and about 100 kilobytes of flash memory. In order to reduce the energy consumption, low performance processors are used on sensor boards. In most cases it is an 8-bit

microcontroller. Therefore, the performance and speed is very limited. Wireless communication has naturally a weak data throughput. Additionally problems with packet failures, packet loss, and collisions lead to increased time delays.

## 8.1 LocSens - Indoor Location Tracking Using Wireless Sensors

LocSens is based on an active environment, where room sensors, attached at fixed positions, communicate with mobile sensors carried by users. In order to determine an exact location, it is necessary to perform distance measurements to multiple points. The test bed in LocSens consist of three rooms of about 80  $m^2$  together, where we placed three sensor boards forming the basic infrastructure. A fourth board is worn by the user, who moves through the environment. Consciously, we use a passive infrastructure, i.e. the room nodes act only as receiver for signals sent by the mobile sensor board. This has the advantage that chronological assignments of the signals onto actual positions become easier.

With this configuration, we recorded a reference model consisting of more than 30.000 data sets. We defined 70 measurement points for all rooms. At each point we gathered data for four orientations (north, west, east, south). By recording several data sets at each point, we aimed to outweigh the jitter of the radio transceivers. Figure 8.1 shows a ground plan of the test environment, as well as room sensors and points of measurements.

The transceiver of the sensor board ESB430 provides a value for the receiving level. This value indicates the distance between the current sensor node and the packet sender. A precise location tracking requires a nearly constant receiving level. Therefore, it is necessary to first test the stability of level values. One of the room sensors is attached to a PC, where measurement values can be read out from all sensors. As the first stability test, we sent sequentially radio packets to all room sensor boards and measured the receiving level. These tests ran over several nights to minimize the signal level deviation caused by interferences, e.g. through persons in the rooms. Figure 8.2 shows the level values of a sensor board measured over a full day. The chart shows that values are mostly located in a thin band, which indicates a good stability. But there exist also some irregular deviations.

Besides the room sensor nodes, there is also a mobile user sensor board. Furthermore, signal strength of the user board is measured by room sensor nodes. As shown in Figure 8.1, points of measurement form a raster with a distance of about one meter between each point. Figure 8.3 shows the user carrying mobile sensor node and measurement equipment. The aim was to build up a reference model

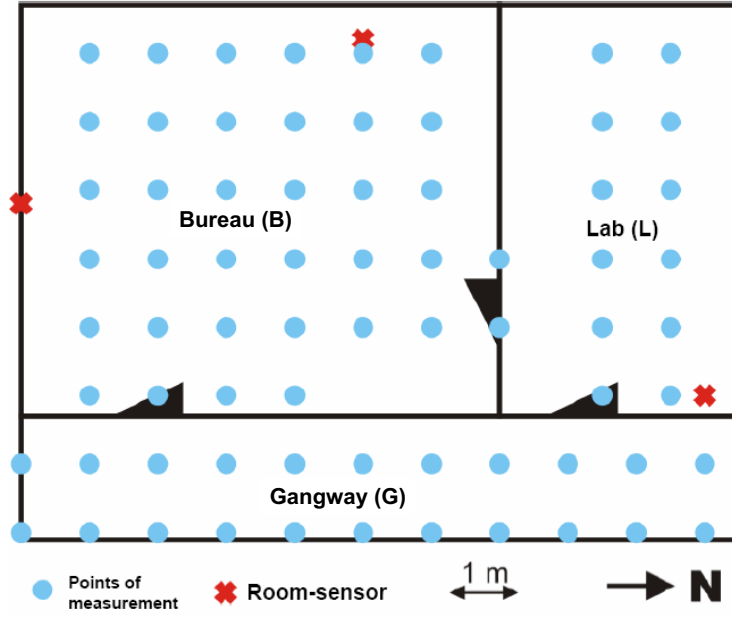


Figure 8.1: Testing environment

that should be used for location estimation performing different algorithms. For each point the coordinates, user's line of sight (N, E, S, W), and receiving level of all three room nodes were stored in a database. In order to have a reliable basis, 120 data points for each line of sight were gathered, achieving 33600 data points altogether.

The location estimation algorithm compares current receiving levels with the data records in the database. In order to optimize the estimation, we implemented several algorithms. The first approach is the *randomized evaluation*, which is actually not practical, but often used as comparison to other approaches. Randomized evaluation chooses randomly  $k$  neighboring points  $r_1, \dots, r_k$  from the reference database. The position  $P$  is calculated by interpolating between these points with fixed weights of  $\frac{1}{k}$ :

$$P = \sum_{i=1}^k \frac{1}{k} r_i \quad (8.1)$$

As further approach for location estimation, we chose *nearest neighbor in signal strength space (NNSS)*. This algorithm differs from the previous one, because it considers  $k$  points  $n_1, \dots, n_k$  from the reference database, which have the lowest Euclidian distance to the current measured receiving level. Position  $P$  can be calculated in the same way by interpolation with weights of  $\frac{1}{k}$ .

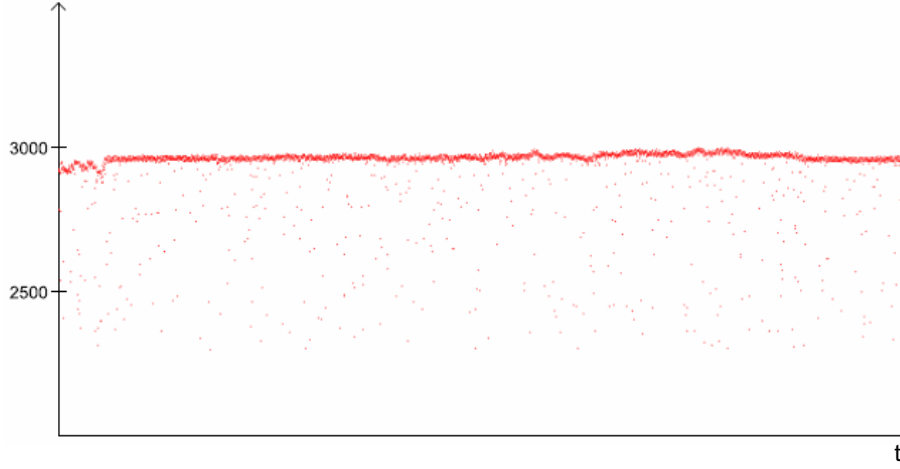


Figure 8.2: Receiving level of a sensor board considered over time

The disadvantage of both previous algorithms is that fixed weights are used. In order to optimize the search for appropriate neighboring points, we implemented a *statistical approach (STAT)* that uses the density of a multidimensional normal distribution as an auxiliary function:

$$f(\hat{x}) = \left(\frac{1}{\sqrt{2\pi}}\right)^n \frac{1}{\sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(\hat{x}-\hat{\mu})^T \Sigma^{-1}(\hat{x}-\hat{\mu})} \quad (8.2)$$

where:

$n \in \mathbb{N}$

$\hat{x} \in \mathbb{R}^n$

$\hat{\mu}$  expectation vector,  $\mu_i = E(X_i)$

$\Sigma$  variance-covariance matrix,  $\sigma_{ii} = Var(X_i)$ ,

$\sigma_{ij} = Cov(X_i, X_j)$  for  $i \neq j$

In the present case with three room nodes, there is a three-dimensional random variable  $X(n = 3)$ , which describes the receiving level at the three room sensor boards. For each reference point  $p$ , the values of  $\hat{\mu}_p$  and  $\Sigma_p$  are calculated. Moreover, the density  $f_p$  can be determined at each reference point  $p$ .

The  $k$  reference points  $s_1, \dots, s_k$  with the highest density  $d_1, \dots, d_k$  are chosen for the estimation of the current position, in the way that weights  $w_1, \dots, w_k$  are calculated as follows:

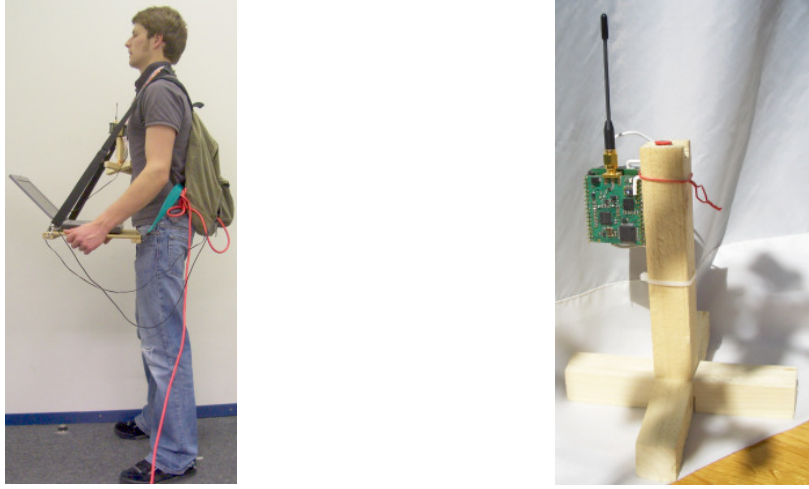


Figure 8.3: Mobile user with measurement equipment and room sensor boards

$$w_i = \frac{d_i}{\sum_{j=1}^k d_j} \forall i = 1..k \quad (8.3)$$

Finally the position  $P$  can be calculated as:

$$P = \sum_{i=1}^k w_i s_i \quad (8.4)$$

## 8.2 Evaluation

Based on the algorithms described in the previous section, we performed several tests, where the location was calculated at different points in the rooms. Standing at a specific position, three neighboring points were chosen for interpolation. Figure 8.4 visualizes the percentiles for all three algorithms (NNSS, STAT, rand), i.e. the percentage of points where the difference between calculated and real position is  $x$  meters (e.g. %25 of the calculated positions have a deviation of 0-1.5 m from the real position). The results show that NNSS and STAT achieve considerably better results than the randomized algorithm, whereas NNSS provides best values.

For the first enhancement, we examined the impact of neighbors by increasing the number of considered neighboring points. Figure 8.5 describes NNSS performed



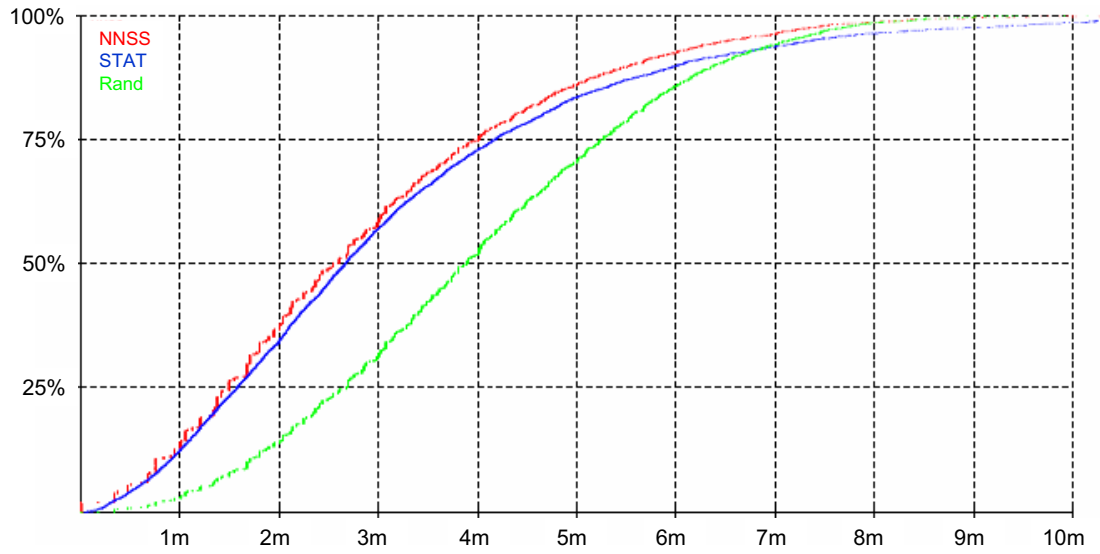


Figure 8.4: Percentile for NNSS, STAT, and random evaluation using 3 neighbors

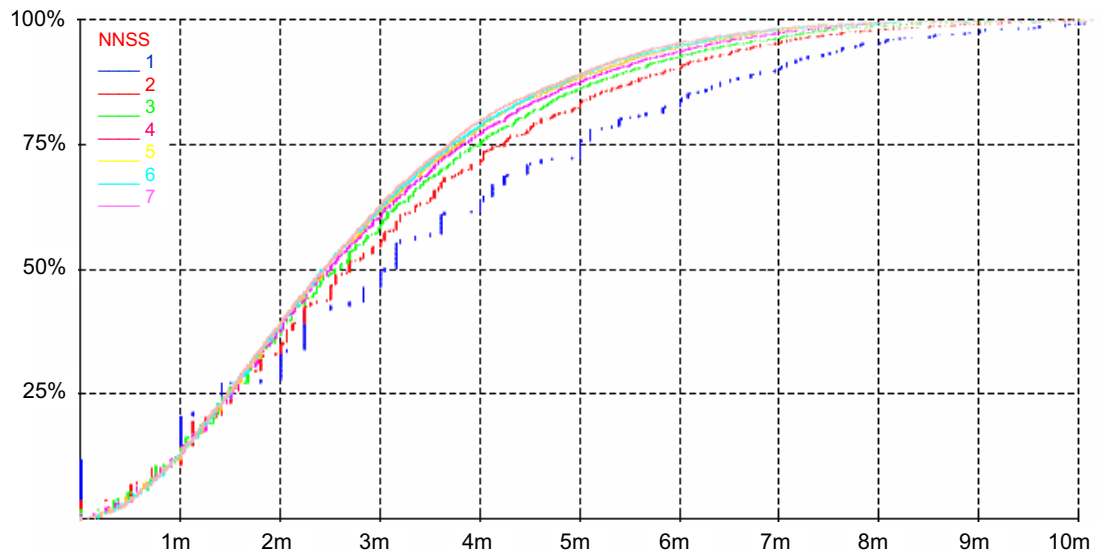


Figure 8.5: Percentile for NNSS using 1-7 neighbors

from 1 to 7 neighbors. Using two neighbors instead of only one provides a clear improvement. In the 90th percentile the deviation decreases by 1.10 m. This value is enhanced by 0.36 m when three neighbors are considered. With more than three neighbors the improvement is not much better. STAT presents similar results. Therefore, it makes only sense to consider at most four neighbors.

In order to examine the impact of the reference data on the precision of location, we performed further tests, changing the number of considered reference points. Decreasing the number of reference points leads to better performance, because the lookup for neighbors takes less time. The results in Figure 8.6 show that even with nearly half of the reference points, STAT provides acceptable values, and NNSS behaves similarly. Also the number of data sets at each reference point can be decreased without losing much location precision.

### 8.2.1 First Optimizations

Nevertheless, it is important to choose the right reference points for the calculation. If the user stands near a wall, it is not reasonable to choose reference points from the other side of the wall. In order to ensure this, we optimized the reference database by storing the current room for each point of measurement. This was easy to realize using the infrared sender/receiver of the sensor board. Figure 8.7 shows that both approaches (NNSS and STAT) provide much better results when additional sensor information is considered.

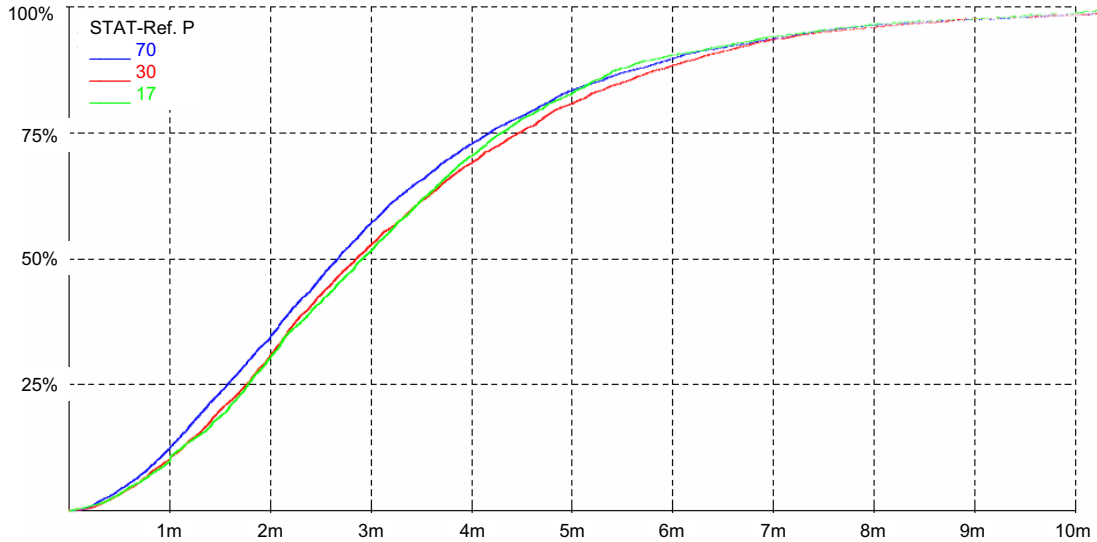


Figure 8.6: Percentile for STAT with different numbers of reference points (70, 30, 17)

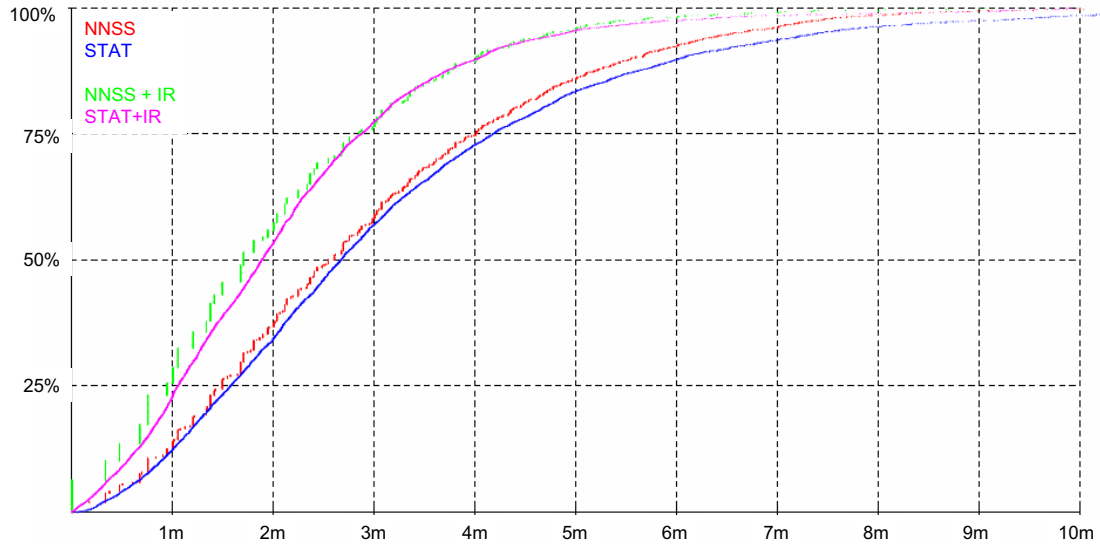


Figure 8.7: Percentile for NNSS and STAT with/without IR information

### 8.2.2 Location Tracking

After examining possible optimizations of location estimation, we performed tests with LocSens where we tracked a moving user. Location tracking is performed by continually calculating timely related data sets. Figure 8.8 shows the actual path of the user through the rooms.

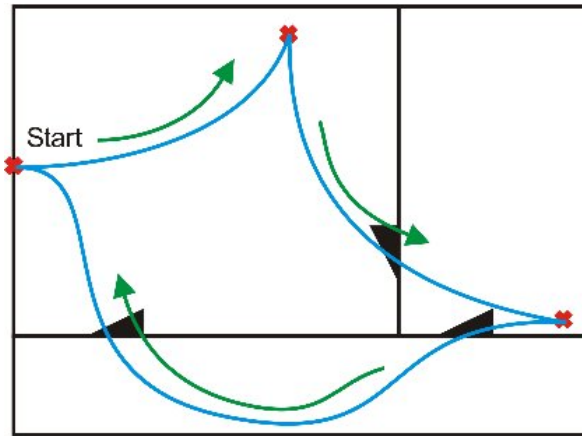


Figure 8.8: Tracking a moving user with LocSens

We chose the STAT algorithm for location tracking. In order to gain fast calculation of current positions, only four data sets for each reference point are

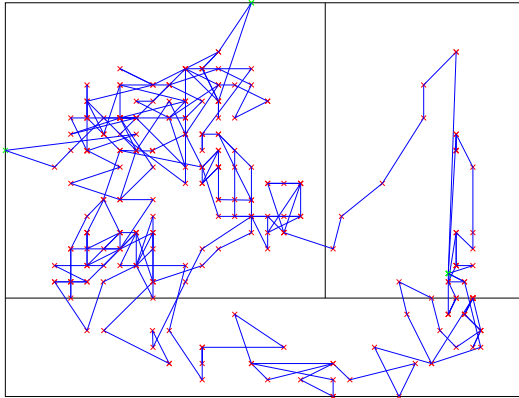


Figure 8.9: Calculated path of user with STAT

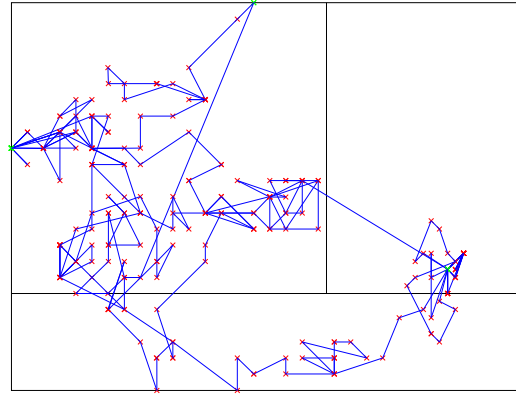


Figure 8.10: Calculated path of user after optimization

considered. This ensures a real-time tracking of the user. Figure 8.9 visualizes the calculated positions of the user moving along the path shown above.

For some points, there are position changes of several meters. Even changes through walls are calculated incorrectly. For further optimizations, we could solve these problems. At first, we limited the considered reference points to only points in a specific distance to the last position.

Using the vibration sensor of the user sensor board, it is possible to detect the motion intensity. This information can be used to dynamically adapt the distance variable. If the user walks faster, the range of considered points increases automatically. Furthermore, position changes through walls can be avoided by allowing room changes only near to doors.

Figure 8.10 shows the calculated path after above mentioned optimizations. The tracking precision in the smaller rooms could be clearly enhanced. In the large room there are still position changes of several meters, but the actual path of the user can be approximated.

## 8.3 Optimizations for LocSens

The results showed so far are not convincing if LocSens is applied on real applications. Therefore, we decided to optimize the system by first using enhanced sensor hardware and then applying new algorithms for location estimation and tracking [BKUB09].

In all tests above, LocSens used the ESB430 sensor boards, which are now replaced by the more up-to-date model MSB430. The MSB430 board has a modular

structure, where specific sensor modules can be easily attached. In the basic form, there exist two modules: a basis module (MSB-430T) and a kernel module (MSB-430). The kernel module uses a TI MSP430F1612 microcontroller and a Chipcon CC1020 radio transceiver, both enhancing the performance in comparison to the older board ESB430. Also three sensors are attached on the kernel module: an acceleration sensor, a temperature sensor, and a humidity sensor. Figure 7.2 shows a picture of the MSB430 basis module.

The main disadvantage of NNSS is that it needs a long time to calculate. Therefore, we modified the NNSS algorithm to achieve enhancements in time of calculation and accuracy. The first idea was to use an arithmetic mean of all relevant reference data points instead of regarding 100 reference data sets in NNSS for each position and line of sight. We call this modified algorithm *M-NNSS*. Following equation clarifies the modification:

$$R_{M-NNSS} = \left\{ \begin{pmatrix} M(x) \\ M(y) \\ M(z) \end{pmatrix} \right\} \quad (8.5)$$

where  $M(a) = \frac{1}{100} \sum_{i=1}^{100} a_i$ .

Figure 8.11 visualizes the percentiles for both algorithms (NNSS and M-NNSS), i.e. the percentage of points where the difference between calculated and real position is at most  $x$  meters (e.g. %25 of the calculated positions have a deviation of 0-1.5 m from the real position). The algorithms were performed regarding 1-7 neighboring points. The results show that M-NNSS achieves higher accuracy in upper percentiles, i.e. there are more points with lower deviation.

### 8.3.1 Cluster optimization

In some cases NNSS chooses neighbors which are too far away from the actual position. This leads to miscalculations that can be eliminated by filtering out neighbors with long distances. The cluster optimization approach chooses only the neighbors which have distances lower than a specific value. Figure 8.12 shows the results of the cluster approach compared to NNSS and M-NNSS using both four neighbors. It is clearly seen that cluster approach achieves much better results than NNSS and M-NNSS. In average the accuracy lies at 1.88m using cluster optimization.

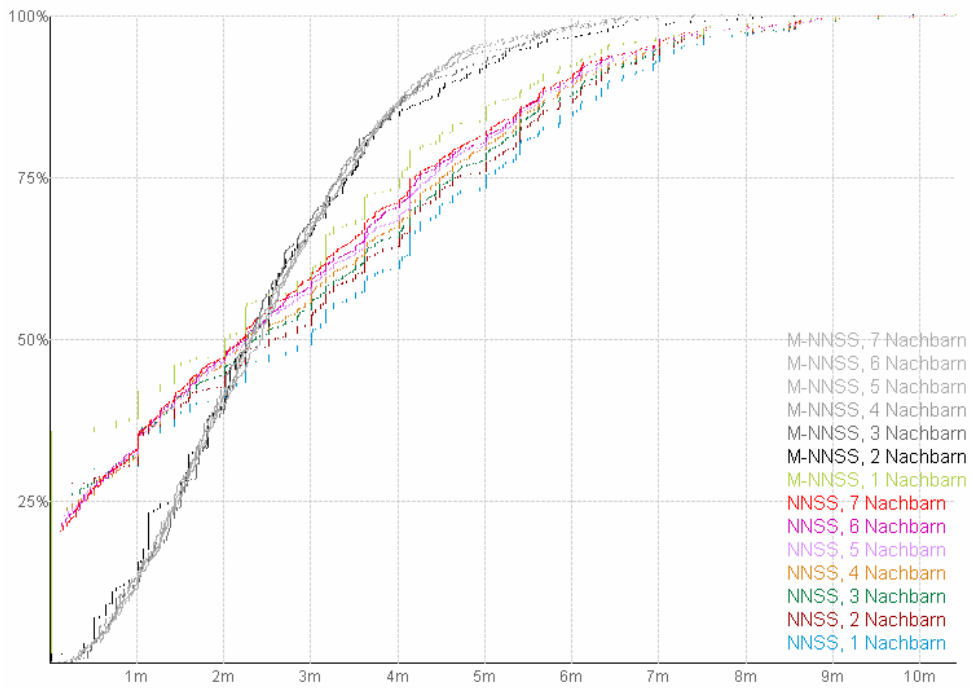


Figure 8.11: Percentile for M-NNSS and NNSS using MSB430 sensor boards

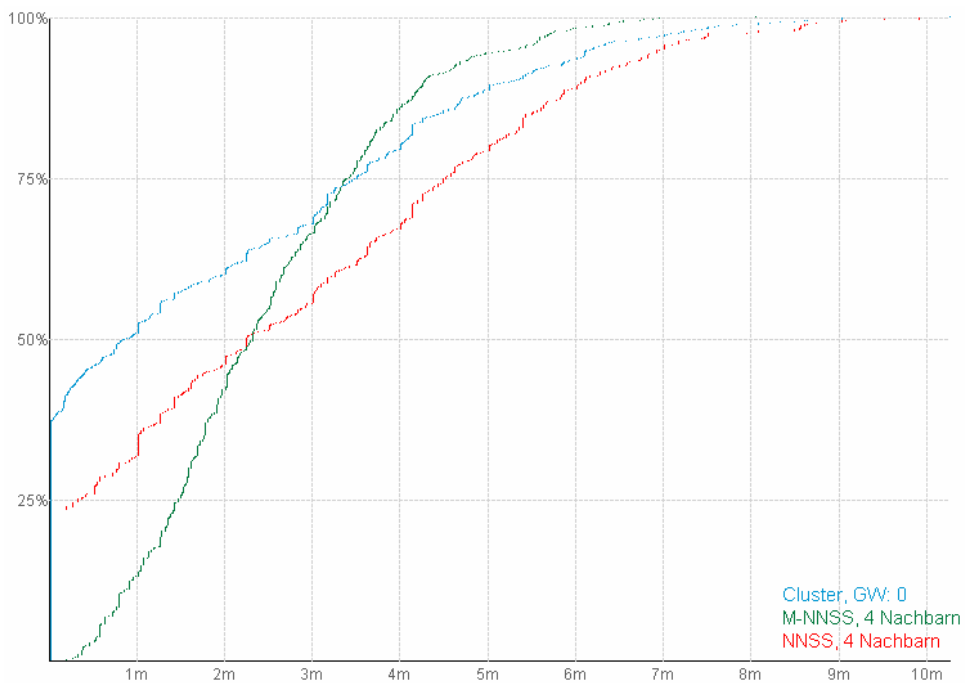


Figure 8.12: Percentile for M-NNSS, NNSS, and Cluster approach

### 8.3.2 Optimization of Room Sensor Positions

The position of room sensor nodes can also influence the location estimation. In order to examine different positions for the three room sensor nodes we established several placement constellations. Figure 8.13 shows four test configurations, where red crosses indicate room sensor nodes.

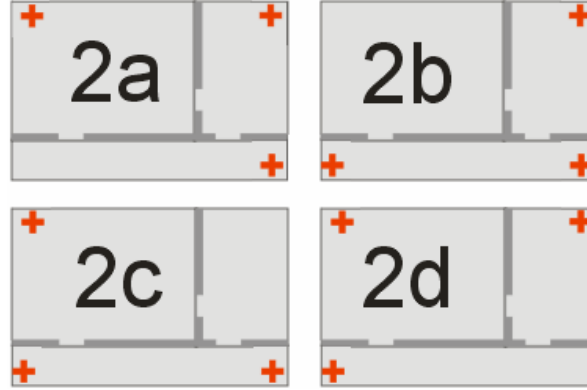


Figure 8.13: Different placements of room sensor nodes

For each constellation we performed the NNSS location estimation algorithm. The results are shown in Figure 8.14. The values for all four constellations are almost parallel, whereas constellation *2d* achieves the best results. We also performed M-NNSS and cluster algorithms for all placements, but the differences between the constellations were unmentionable low.

The next logical step would be to use four room sensor nodes instead of only three. This implies an increase of processing time due to additional calculations. But as you can see in Figure 8.15 all three algorithms achieve better results. The cluster approach gain  $0.68m$  lower accuracy in average to  $1.20m$ , which is obviously worth the additional calculations.

In all above test scenarios the user was carrying the mobile sensor node at his chest. Since the body blocks radio signals, the received signal level in some positions would be corrupted. In order to examine this fact we placed the mobile sensor node on top of user's head. Figure 8.16 shows a picture of the user. In this optimal scenario we could achieve the best results for all algorithms. Table 8.1 describes some chosen percentiles for NNSS, M-NNSS, and cluster algorithms. Unfortunately, carrying the sensor node on top of the head is inconvenient for a person. Therefore this is not a realistic scenario. Nevertheless, this evaluation clarified the impact of interferences in wireless communication.

Table 8.2 gives an overview of several location estimation techniques compared

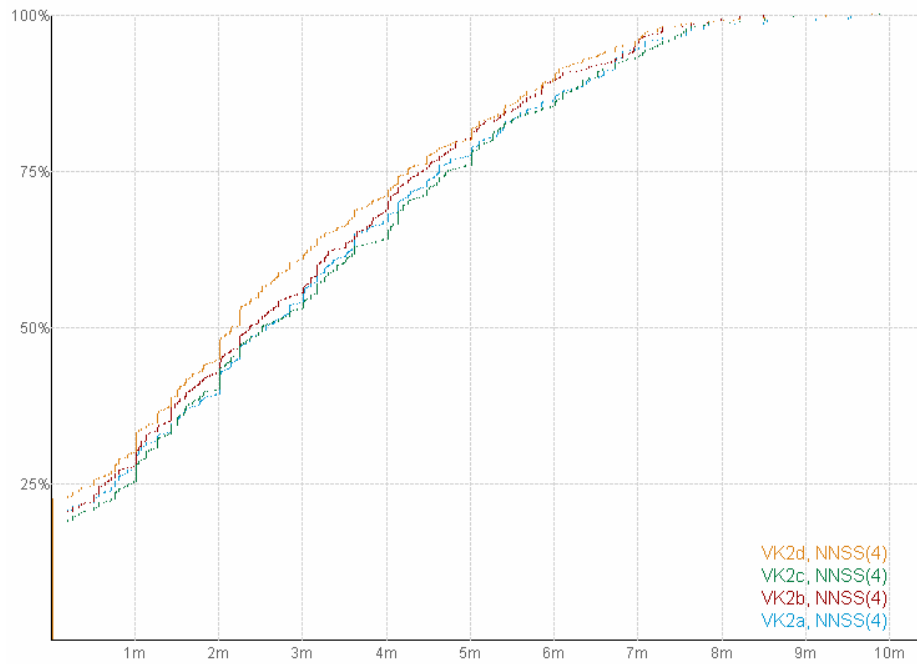


Figure 8.14: Percentile for NNSS in different room node placements

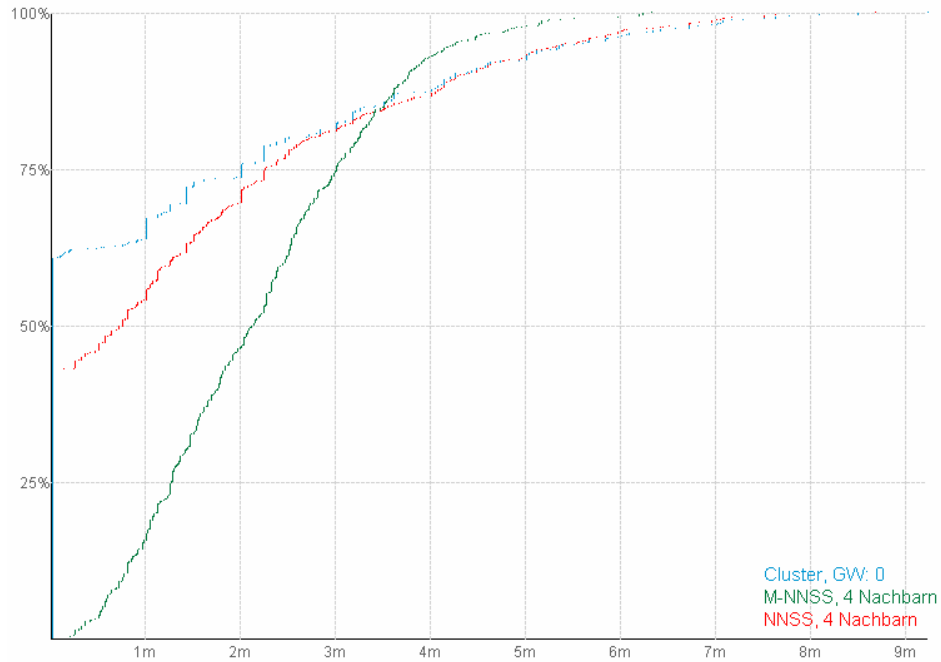


Figure 8.15: Percentile for NNSS, M-NNSS, and cluster using four room sensor nodes





Figure 8.16: Mobile sensor node on top of user's head

Table 8.1: Chosen percentiles for NNSS, M-NNSS, and cluster where mobile node is on user's head

	<b>25%</b>	<b>50%</b>	<b>75%</b>	<b>100%</b>	<b>average</b>
NNSS	0.00	0.50	1.91	9.55	1.18
M-NNSS	1.12	1.68	2.26	5.34	1.82
Cluster	0.00	0.00	1.80	10.26	1.05

with LocSens. As seen in this table LocSens achieves very good accuracy values with very low installation costs.

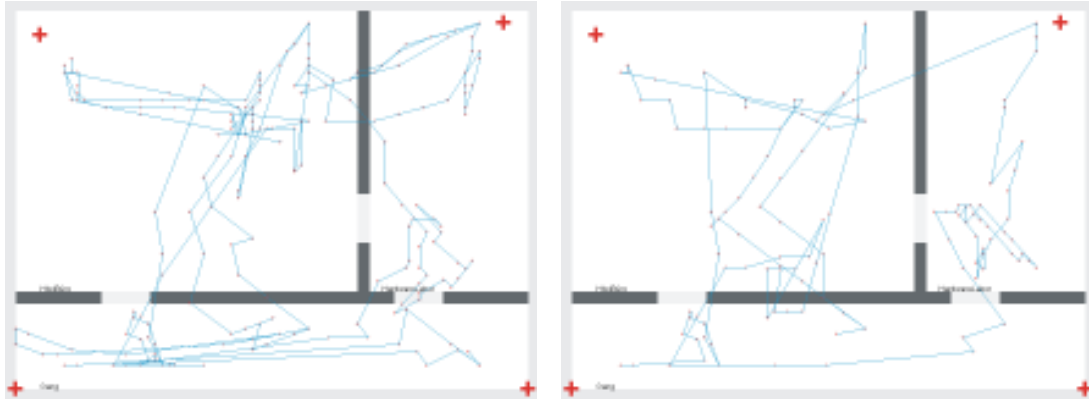
### 8.3.3 Optimization of Location Tracking

Since four room sensor nodes achieved best results in location estimation, we also used four room nodes for tracking. The user carries again a mobile sensor node on his chest. Room nodes request periodically location messages from the mobile node. Using signal strength level each node estimates the actual location of the user based on NNSS algorithm. Figure 8.8 shows the actual path of the user through three rooms.

Table 8.2: Comparison of location estimation techniques

System	Technology	Range (indoors)	Accuracy	Size of test environment
Active Badge	Infrared	< 6m	exact room	unknown
RADAR	WLAN	25m - 50m	2.75m	44m x 23m
LANDMARC	active RFID	50m	1m	9m x 4m
LocSens (ESB430)	RF	< 20m	2.90m	10m x 8m
LocSens (MSB430)	RF	< 20m	1.05m	10m x 8m

The first optimization was to use sliding windows, i.e. we collected a specific number of signal measurements and calculated user's location over the means of collected data sets. Figure 8.17 shows enhancements of location tracking using sliding windows.  $W(x)$  indicates calculation based on  $x$  collected data sets. Increasing the number of data sets achieves better results, but the calculation needs more time. With this first optimization you can nearly see the path of the user, but still there are some miscalculated positions. In order to eliminate them we performed further optimization steps.

Figure 8.17: Location tracking using sliding windows:  $W(50)$  and  $W(100)$ 

Using the acceleration sensor of the user sensor board, it is possible to detect the motion intensity. This information can be used to dynamically adapt the size of the sliding window. If the user walks faster, the range of considered points increases automatically.

As you can see in Figure 8.17 there are several calculated location changes going through walls. This unlikely situations can be avoided by allowing room changes only near to doors. For this reason we adapted our algorithm to consider data

sets from other rooms only if the preceding location was in vicinity of a door. Figure 8.18 illustrates the results after above mentioned optimizations. Still there are some miscalculated locations, but you can easily recognize the path of user across the rooms.

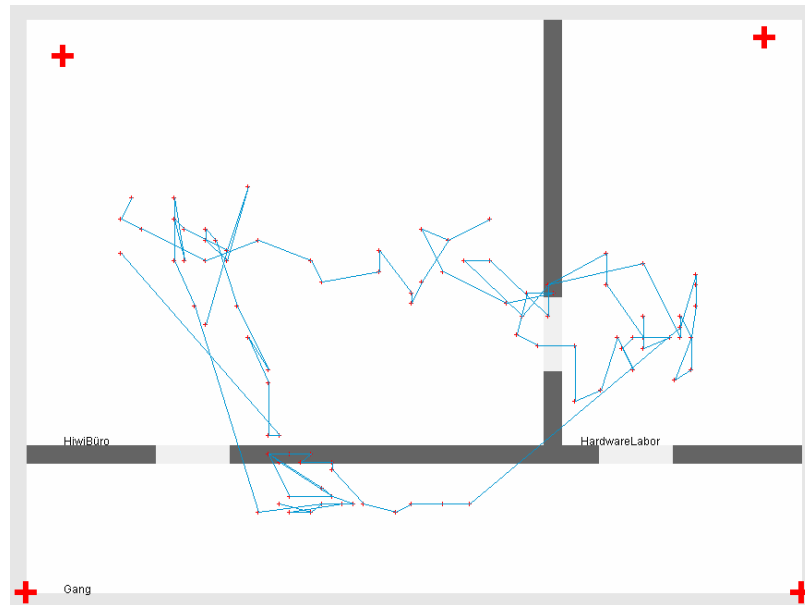


Figure 8.18: Calculated path of user with several optimizations

## 9 Conclusion

This last chapter summarizes the main achievements of this thesis and provides an outlook to further research that may lead to the ultimate goal of tiny and very low cost energy-efficient wireless sensor nodes that can be used for security related applications.

We described a new system architecture for WSNs that allows applications to access parameters on different levels to control energy-saving and security. These features are included in a middleware providing interfaces on application level. This abstraction opens the possibility to integrate different sensor hardware into the WSN in an easy way. The middleware offers functionalities to realize communication, to adjust network topology, to update and reprogram applications, to localize and track sensor nodes, and to secure data and communication. All these functionalities are performed in an energy-efficient way to overcome performance needs for protocol overheads and encryption.

In terms of local communication, this thesis presented the *Energy-saving Token Ring Protocol ESTR* for wireless sensor networks. ESTR uses sleep phases to decrease the energy consumption. Only the node that holds the token can receive messages from other nodes. This reduces idle-listening and overhearing. ESTR provides to adjust the maximal permitted ring size, offering a flexible method to control throughput and energy consumption. Furthermore, the token includes energy information of preceding nodes that allows self-optimization of the ring by dynamically adapting the sleep time of each node. In detailed simulations, ESTR was compared to related MAC protocols. ESTR achieved the best energy results. The probability of collisions in ESTR is less than in other protocols and ESTR offers a dynamic energy optimization in the overall network, increasing network life-time. In further evaluations we measured communication latency. Although the advantage of ESTR lies in energy-efficiency, it achieves reasonable latency results. We also proved the usability of ESTR in scenarios with real sensor boards.

Cluster architectures offer a good basis for scalable and energy-efficient protocols. Using hierarchy of cluster-heads and sensor nodes, it is possible to limit the range of each node and to exploit multi-hop communication. This thesis presented the Variable Ranges protocol for WSN that allows applications to adjust the topology. By dynamically adapting the range of each node, the network can be

established with low complexity, but still with high connectivity. Since nodes do not sent messages with full transmission power, the energy consumption decreases considerably. Evaluations showed that VR protocol achieved an extended lifetime of the overall sensor network without decreasing the connectivity.

In the field of code mobility this thesis presented the *Ubiquitous Mobile Agent System for wireless sensor networks - UbiMASS*. UbiMASS is very light-weighted so that it can run on tiny devices like sensor boards. It is designed using a modular architecture in order to ease adaption for other hardware and to offer a convenient way for programming new services in the form of agents. Using the wireless link agents can migrate to remote nodes. The UbiMASS communication layer is responsible for the reliable transfer of agents. In order to load and bind agent code dynamically, UbiMASS provides the ELF loader that is based on a standard approach. The agent can access sensor and actuator information using interfaces of the underlying middleware. The migration engine offers agents the possibility to autonomously decide if they want to move to other nodes. We proved the usability of UbiMASS in scenarios with real sensor boards. Compared to other systems the reprogramming of nodes can be done in a more flexible and dynamic way using UbiMASS.

The main focus of this thesis lies on security. We presented here *SecSens*, a multi-level security architecture for wireless sensor networks. SecSens combines several security approaches on different system levels in order to provide high protection. SecSens contains four components, which interact with each other: authenticated broadcasts, key management, routing, and en-route filtering. We implemented a simulation tool, where huge network sizes can be established. Evaluation results show that SecSens can resist DoS and insider attacks limiting failures to a local area. Infiltrated false sensor data can be refused with a high probability. We also demonstrated the feasibility of SecSens by building a real sensor network environment with two different kinds of sensor boards.

As last feature of the proposed system architecture we presented *LocSens*, an indoor location estimation and tracking system based on wireless sensors. Since sensor boards are produced with low costs, the usage of wireless sensors minimizes installation cost of the overall system. LocSens was implemented with different location estimation algorithms, namely NNSS, M-NNSS, and cluster approach. We evaluated the impact of several modifications on the test environment and reference data, which resulted in enhancements of system performance. Also the usage of additional sensor information increased precision of calculation. Especially, location tracking can benefit from data about movement intensity of the user. Compared to other indoor location systems LocSens achieves acceptable results in location estimation and real-time location tracking with considerably low installation costs.

In summary, this thesis provides a novel energy-efficient and secure system architecture for wireless sensor networks. However, there still exist open problems or application scenario specific problems. It is very complicated to detect attacks especially if it is an insider attack. Possible intrusion detection mechanisms could be used also in wireless sensor network in future. Another promising area for further research is the use of tamper-resistant hardware in WSNs. Although tamper-resistant hardware cannot be used in the majority of application scenarios, it is indispensable for most high-security scenarios to prevent insider attacks. Since these scenarios may also have different requirements, new application-specific protocols are needed. As a promising technology to close the energy gap are energy scavenging techniques. These approaches could be used to extend wireless sensor network by automatically reloading batteries or to run nodes completely without batteries. There are still many open issues and problems to be solved in the construction of energy scavenging systems. More sophisticated micro electro-mechanical systems (MEMS) will allow to extract more energy at a limited size of the sensor but a single energy source such as coming from light, vibration, temperature gradients etc. will always be problematic because it is not reliably available in most of the applications. Combining different types of energy scavenging systems will make the energy source more reliable, but causes an increase of complexity and cost.



# Bibliography

- [AKWK04] I. Avramopoulos, H. Kopayashi, R. Wang, and A. Krishnamurthy. Highly Secure and Efficient Routing. *IEEE INFOCOM*, 1:197–208, 2004.
- [Bag11] Faruk Bagci. Energy-aware Clustering with Variable Ranges in Wireless Sensor Networks. In *Proceedings of the Fifth International Conference on Sensor Technologies and Applications (SENSORCOMM '11)*, Nice/Saint Laurent du Var, France, August 2011.
- [BKJS11] Faruk Bagci, Alaa Khalifeh, Georg Jung, and Christian Sturm. Variable Ranges Security Protocol for Wireless Sensor Networks. In *Proceedings of the 2011 International Conference on Wireless Networks (ICWN '11)*, Las Vegas, USA, July 2011.
- [BKS<sup>+</sup>08] Faruk Bagci, Florian Kluge, Benjamin Satzger, Andreas Pietzowski, Wolfgang Trumler, and Theo Ungerer. *Experiences with a Smart Office Project*. Mobile Intelligence: Mobile Computing and Computational Intelligence by Laurence T. Yang, Wiley-Interscience, 2008.
- [BKSU09] Faruk Bagci, Florian Kluge, Benjamin Satzger, and Theo Ungerer. Towards Indoor Location Estimation and Tracking with Wireless Sensors. In *IEEE International Symposium on Intelligent Signal Processing (WISP '09)*, Budapest, Hungary, August 2009.
- [BKUB09] Faruk Bagci, Florian Kluge, Theo Ungerer, and Nader Bagherzadeh. Optimizations for LocSens - An Indoor Location Tracking System using Wireless Sensors. *International Journal of Sensor Networks (IJSNET)*, 6(3/4), 2009.
- [BP00] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-building RF-based User Location and Tracking System. In *INFOCOM (2)*, pages 775–784, 2000.
- [BPC<sup>+</sup>07] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications* 30, pages 1655–1695, 2007.



- [BTn07] BTnode Platform. <http://www.btnode.ethz.ch/Main/Overview>, 2007.
- [BUB08a] Faruk Bagci, Theo Ungerer, and Nader Bagherzadeh. ESTR - Energy Saving Token Ring Protocol for Wireless Sensor Networks. In *Proceedings of the International Conference on Wireless Networks (ICWN '08)*, Las Vegas, USA, July 2008.
- [BUB08b] Faruk Bagci, Theo Ungerer, and Nader Bagherzadeh. LocSens - An Indoor Location Tracking System using Wireless Sensors. In *Proceedings of the First IEEE International Workshop on Sensor Networks (SN 2008)*, Virgin Islands, USA, August 2008.
- [BUB09] Faruk Bagci, Theo Ungerer, and Nader Bagherzadeh. SecSens - Security Architecture for Wireless Sensor Networks. In *Proceedings of the Third International Conference on Sensor Technologies and Applications (SENSORCOMM '09)*, Athens, Greece, June 2009.
- [BUB10] Faruk Bagci, Theo Ungerer, and Nader Bagherzadeh. Multi-level Security in Wireless Sensor Networks. *International Journal of Advances in Software*, 4(6), 2010.
- [BWSU10] Faruk Bagci, Julian Wolf, Benjamin Satzger, and Theo Ungerer. UbiMASS - Ubiquitous Mobile Agent System for Wireless Sensor Networks. In *Proceedings of the Third IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC '10)*, Newport Beach, USA, June 2010.
- [BWUB09] Faruk Bagci, Julian Wolf, Theo Ungerer, and Nader Bagherzadeh. Mobile Agents for Wireless Sensor Networks. In *Proceedings of the International Conference on Wireless Networks (ICWN '09)*, Las Vegas, USA, July 2009.
- [CKM00] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and Approaches for Distributed Sensor Security. Technical Report 00-010, NAI Labs tech. rep., 2000.
- [CPS04] Haowen Chan, Adrian Perrig, and Dawn Song. Key distribution techniques for sensor networks. *Wireless sensor networks*, pages 277–303, 2004.
- [Cro07] Crossbow Technology Inc. Stargate datasheet. [http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/Stargate\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Stargate_Datasheet.pdf), 2007.
- [DFEV06] Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. Run-Time Dynamic Linking for Reprogramming Wireless Sensor

- Networks. In *Proceeding of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006.
- [DGV04] Adam Dunkels, Bjoern Groenvall, and Thiemo Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.
- [DHM06] Jing Deng, Richard Han, and Shivakant Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, 2006.
- [DLWW04] Zhenhua Deng, Yan Lu, Chunjiang Wang, and Wenbo Wang. E<sup>2</sup>WTRP: An Energy-Efficient Wireless Token Ring Protocol. In *Proceeding of the IEEE conference on Personal, Indoor, and Mobile Radio Communications*, pages 398–401, Barcelona, Spanien, 2004.
- [ELF95] ELF. Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2. <http://refspecs.freestandards.org/elf/elf.pdf>, 1995.
- [ELSV04] M. Ergen, D. Lee, R. Sengupta, and P. Varaiya. WTRP - Wireless Token Ring Protocol. *IEEE Transactions on Vehicular Technology*, 53(6):1863–1881, November 2004.
- [EnO07] EnOcean Technology. <http://www.enocean.com/en/>, 2007.
- [FRL05] C.-L. Fok, G.-C. Roman, and C. Lu. Mobile Agent Middleware for Sensor Networks: An Application Case Study. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05)*, pages 382–387, April 2005.
- [FRL06] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Agilla: A Mobile Agent Middleware for Sensor Networks. Technical Report WUCSE-2006-16, Department of Computer Science and Engineering, Washington University in St. Louis, 2006.
- [GACE04] D. Ganesan, Y.Yu A. Cerpa, and D. Estrin. Networking issues in wireless sensor networks. *Journal of Parallel and Distributed Computing (JPDC)*, Special issue on Frontiers in Distributed Sensor Networks, 2004.
- [GI04] S. Goel and T. Imielinski. Etiquette Protocol for Ultra Low Power Operation in Sensor Networks. Technical Report DCS-TR-552DCS-TR-552, CS Dept., Rutgers University, 2004.
- [GLvB<sup>+</sup>03] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric

- Brewer, and David Culler. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (PLDI)*, pages 1–11, ACM, 2003.
- [HC02] Jason Hill and David Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.
- [HH04] L. V. Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In *Proceedings of INSS 004*, June 2004.
- [HHKK04] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The platforms enabling wireless sensor networks. *Commun. ACM*, 47(6):41–46, 2004.
- [HHS<sup>+</sup>02] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The Anatomy of a Context-aware Application. *Wirel. Netw.*, 2(2/3):187–197, 2002.
- [HMH06] M. Abdul Hamid, M. Mamun, and Choong Seon Hong. Routing Security in Sensor Network: HELLO Flood Attack and Defense. *IC-NEWS (2006)*, pages 77–81, 2006.
- [HPJ03] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. *IEEE INFOCOM*, 1:1976–1986, 2003.
- [HSW<sup>+</sup>00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.
- [IEE85] IEEE. IEEE CS, Token Ring Access Method and Physical Layer Specifications. ANSI/IEEE Standard 802.5, 1985.
- [IEE06] IEEE Computer Society. IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks, part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>, 2006.
- [JNSP04] E. Shi J. Newsome, D. Song, and A. Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. *International Symposium on Information Processing in Sensor Networks*, 1:259–268, 2004.

- [KLP02] Chris Karlof, Yaping Li, and Joe Polastre. Arrive: Algorithm for robust routing in volatile environments. Technical Report UCB/CSD-03-1233, University of California at Berkeley, May 2002.
- [KSW04] C. Karlof, N. Shastry, and D. Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of SenSys 04*, Baltimore, MD, November 2004.
- [KW03] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2-3):293–315, 2003.
- [LBR<sup>+</sup>02] Andrew M. Ladd, Kostas E. Bekris, Algis Rudys, Lydia E. Kavraki, Dan S. Wallach, and Guillaume Marceau. Robotics-based Location Sensing Using Wireless Ethernet. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 227–238, ACM Press, 2002.
- [LH05] Y.W. Law and P. J. M. Havinga. How to secure sensor network. In *Proceedings of the 2005 International Conference on Sensor Networks and Information Processing*, pages 89–95, 2005.
- [LN04] Donggang Liu and Peng Ning. Multilevel  $\mu$ tesla: Broadcast authentication for distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(4):800–836, 2004.
- [MNP04] A. Mishra, K. Nadkarni, and A. Patcha. Intrusion Detection in Wireless Ad Hoc Networks. *IEEE Wireless Communications*, 11(1):48–60, 2004.
- [NLLP04] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. Landmarc: Indoor Location Sensing Using Active RFID. *Wirel. Netw.*, 10(6):701–710, 2004.
- [PSC05] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.
- [PST<sup>+</sup>02] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.
- [RY06] Xiuli Ren and Haibin Yu. Security Mechanisms for Wireless Sensor Networks. *IJCSNS International Journal of Computer Science and Network Security*, 6(3):155–161, 2006.

- [SACO06] Rajeev Shorey, A. Ananda, Mun Choon Chan, and Wei Tsang Ooi. *Mobile, Wireless, and Sensor Networks Technology, Applications, and Future Directions*. John Wiley & Sons Ltd, 2006.
- [SBGP04] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. Tracking Moving Devices with the Cricket Location System. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 190–202, 2004.
- [Sca07] ScatterWeb Homepage. <http://www.scatterweb.com>, 2007.
- [SCL04] S. Sancak, E. Cayrici, V. Coskun, and A. Levi. Detecting and Defending Against Spam Attacks in Wireless Sensor Networks. In *Proceedings of IEEE International Conference on Communications*, pages 3668–3672, 2004.
- [SP04] E. Shi and A. Perrig. Designing Secure Sensor Networks. *Wireless Communication Magazine*, 11(6):38–43, 2004.
- [Sun] Sun Microsystems, Inc. Project Sun SPOT. <http://www.sunspotworld.com>.
- [TBPU03] Wolfgang Trumler, Faruk Bagci, Jan Petzold, and Theo Ungerer. Smart Doorplate. In *First International Conference on Appliance Design (1AD)*, Bristol, GB, May 2003. Reprinted in *Pers Ubiquit Comput* (2003) 7: 221–226.
- [The07] The Network Simulator - NS-2. [http://nsnam.isi.edu/nsnam/index.php/Main\\_Page](http://nsnam.isi.edu/nsnam/index.php/Main_Page), 2007.
- [UAJP02] J. Undercoffer, S. Avancha, A. Joshi, and J. Pinkston. Security for sensor networks. *CADIP Research Symposium*, 2002.
- [Uni] University of California Berkeley: TinyOS. <http://www.tinyos.net/>.
- [vDL03] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1567–1576, Los Angeles, California, USA, November 2003.
- [Wag04] D. Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 78–87, ACM Press, New York, 2004.
- [WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The Active Badge Location System. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

- [WLSC06] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary. Wireless Sensor Network Security: A Survey. *Security in Distributed, Grid, and Pervasive Computing*, 2006.
- [WS02] Anthony D. Wood and John A. Stankovic. Denial of Service in Sensor Networks. *IEEE Computer*, pages 54–66, October 2002.
- [YHE02] Wei Ye, John Heidemann, and Deborah Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1567–1576, New York, USA, June 2002.
- [YHE04] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, June 2004.
- [YLLZ05] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Self-organizing Distributed Collaborative Sensor Networks*, 23(4):839–850, April 2005.
- [YYY<sup>+</sup>05] Hao Yang, Fan Ye, Yuan Yuan, Songwu Lu, and William Arbaugh. Toward resilient security in wireless sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 34–45, New York, NY, USA, 2005. ACM Press.
- [ZSJ03] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 62–72, New York, NY, USA, 2003. ACM Press.
- [ZSJN04] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. *2004 IEEE Symposium on Security and Privacy*, 00:259–271, 2004.



# List of Figures

1.1	Distributed System Architecture for Wireless Sensor Networks . .	4
2.1	Basic sensor node components . . . . .	8
2.2	Berkeley MICA2 Mote . . . . .	10
2.3	Sun SPOT Mote . . . . .	11
4.1	Eulerian cycle . . . . .	29
4.2	Multiple interconnected rings . . . . .	30
4.3	Token hand-over and message exchange . . . . .	31
4.4	State chart of the ESTR-MAC-level . . . . .	32
4.5	Sections of the awake phase . . . . .	32
4.6	Energy efficiency with 20 nodes . . . . .	34
4.7	Energy efficiency with 100 nodes . . . . .	35
4.8	Energy efficiency with 500 nodes . . . . .	35
4.9	Energy minimum in the sensor network . . . . .	37
4.10	Energy quotient . . . . .	37
4.11	Ping-pong between two nodes . . . . .	38
4.12	Time for a ping-pong between far nodes . . . . .	39
4.13	Time for a ping-pong between close nodes . . . . .	40
4.14	Real-life scenario with one ring . . . . .	41
4.15	Real-life scenario with two interconnected rings . . . . .	41
4.16	Energy consumption in real-life . . . . .	41
5.1	Basic sensor network architecture . . . . .	43
5.2	Range adjustment in VR protocol . . . . .	45
5.3	Initial phases of VR protocol: a) neighbor search b) TryToConnect	46
5.4	The Simulator GUI . . . . .	47
5.5	Traffic load in relation to signal strength . . . . .	48
5.6	Traffic load for different configurations of VR protocol . . . . .	50
5.7	Connectivity for different configurations of VR protocol . . . . .	50
5.8	Network complexity without (left) and with VR protocol (right) .	51
5.9	Energy consumption for reporting . . . . .	52
6.1	UbiMASS host architecture . . . . .	54
6.2	Different views of an ELF file . . . . .	56
6.3	Agent migration in UbiMASS . . . . .	58



6.4	Example: sending of variable values . . . . .	58
6.5	Access of sensor and actuators of the ESB430 in UbiMASS . . . . .	59
6.6	UDP datagram format in UbiMASS . . . . .	60
6.7	Example of a UbiMASS agent . . . . .	61
6.8	Round trip of a mobile agent in UbiMASS . . . . .	62
6.9	Round trip of two successive mobile agents in UbiMASS . . . . .	63
6.10	Round trip in Contiki . . . . .	63
6.11	Receiving an agent: only transfer time vs loading time . . . . .	64
6.12	Receiving an agent: comparison with reprogramming in Contiki . . . . .	65
6.13	Round trip time subject to distance between sensor nodes . . . . .	65
7.1	Key chain approach . . . . .	77
7.2	Sensor boards: ESB 430/1 and MSB-430 . . . . .	82
7.3	Memory map of SecSens security components . . . . .	84
7.4	Consumed energy for reporting . . . . .	86
7.5	Energy consumption with passive participation . . . . .	86
8.1	Testing environment . . . . .	89
8.2	Receiving level of a sensor board considered over time . . . . .	90
8.3	Mobile user with measurement equipment and room sensor boards . . . . .	91
8.4	Percentile for NNSS, STAT, and random evaluation using 3 neighbors . . . . .	92
8.5	Percentile for NNSS using 1-7 neighbors . . . . .	92
8.6	Percentile for STAT with different numbers of reference points (70, 30, 17) . . . . .	93
8.7	Percentile for NNSS and STAT with/without IR information . . . . .	94
8.8	Tracking a moving user with LocSens . . . . .	94
8.9	Calculated path of user with STAT . . . . .	95
8.10	Calculated path of user after optimization . . . . .	95
8.11	Percentile for M-NNSS and NNSS using MSB430 sensor boards . . . . .	97
8.12	Percentile for M-NNSS, NNSS, and Cluster approach . . . . .	97
8.13	Different placements of room sensor nodes . . . . .	98
8.14	Percentile for NNSS in different room node placements . . . . .	99
8.15	Percentile for NNSS, M-NNSS, and cluster using four room sensor nodes . . . . .	99
8.16	Mobile sensor node on top of user's head . . . . .	100
8.17	Location tracking using sliding windows: $W(50)$ and $W(100)$ . . . . .	101
8.18	Calculated path of user with several optimizations . . . . .	102

# List of Tables

7.1	Memory requirements for SADS . . . . .	83
7.2	Initialization . . . . .	85
8.1	Chosen percentiles for NNSS, M-NNSS, and cluster where mobile node is on user's head . . . . .	100
8.2	Comparison of location estimation techniques . . . . .	101